UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
COMUE UNIVERSITE BRETAGNE LOIRE

Ecole Doctorale N°601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité Informatique
Par

Alban SIFFER

New Statistical Methods for Data Mining
Contributions to Anomaly Detection and Unimodality Testing

Thèse présentée et soutenue à RENNES, le 19 décembre 2019

Rapporteurs avant soutenance :

Tijl DE BIE, Full professor at Ghent University
Arnaud GIACOMETTI, Professeur des Universités à l'Université de Tours

Composition du jury :

Présidente:	Magalie FROMONT, Professeur des Universités à l'Université de Rennes 2
Examinateurs:	Tijl DE BIE, Full professor at Ghent University Arnaud GIACOMETTI, Professeur des Universités à l'Université de Tours Céline ROBARDET, Professeur des Universités à l'INSA de Lyon Anne SABOURIN, Maître de conférence à Telecom ParisTech Christine LARGOUËT, Maître de conférence à AGROCAMPUS OUEST
Directeur de thèse:	Pierre-Alain FOUQUE, Professeur des Universités à l'Université de Rennes 1
Co-directeur de thèse:	Alexandre TERMIER, Professeur des Universités à l'Université de Rennes 1

People think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones.

Donald Knuth

Remerciements

Je souhaite en premier lieu remercier les membres du jury. Malgré une fin d'année dense, je suis heureux que vous ayez accepté d'en faire partie. Je remercie particulièrement Arnaud Giacometti et Tijl de Bie pour leurs commentaires et leurs rapports très positifs à l'égard de ce travail.

Si je devais faire le bilan de ces trois années, il ne pourrait se résumer en une simple phrase tellement j'ai appris.

Merci Pierre-Alain et Alexandre pour votre accompagnement et votre soutien mais surtout pour la vision de la recherche que vous m'avez transmise. Je crois désormais que c'est un savant mélange entre une histoire à raconter, de la science (i.e. des maths) et une bonne confiance en soi. J'ai en effet pu ressentir de nombreuses fois cet éternel doute qu'est la recherche, mais avec expérience et ambition, vous m'avez fait franchir ces obstacles. Christine, je peux évidemment te faire les mêmes compliments. J'y ajoute un grand merci pour ta disponibilité mais également ta tempérance qui a permis un équilibre parfait durant ces trois années. Étrangement, je repense à tous ces fous-rires que nous avons eu tous les quatre en réunion, à croire qu'il ne faut peut-être pas trop se prendre au sérieux pour avancer.

Je remercie évidemment Amossys pour avoir financé cette thèse, et en particulier Frédéric Rémi pour la vision qu'il portait en prémices de celle-ci. Je remercie François qui m'a encadré au tout début de cette aventure et qui m'a réellement insufflé l'esprit de cette thèse. Merci à Fred pour ta confiance et l'autonomie que tu m'as laissée durant la recherche. Je remercie enfin tous mes collègues d'Amossys qui ont pu me côtoyer de près ou de loin et qui ont dû se demander de nombreuses fois ce que je pouvais bien être en

train de chercher. Une mention spéciale au pôle audit dans lequel je suis historiquement immergé. Il y a un peu plus de trois ans, je ne connaissais rien en matière de sécurité informatique et aujourd'hui je dois presque faire illusion à vos côtés.

Avec deux directeurs de thèse, j'ai également eu le privilège de faire partie de deux équipes au sein du laboratoire, EMSEC et LACODAM. Difficile d'en profiter pleinement sans don d'ubiquité! Malgré ma faible présence, j'ai réellement apprécié l'esprit au sein de LACODAM. J'espère que vous m'inviterez un jour à Tahiti et que j'aurai de nouvelles choses à vous raconter.

Ces sont majoritairement les cryptologues d'EMSEC qui ont donc eu le bonheur de compter un analyste dans leurs rangs. J'ai tellement appris auprès d'eux. Je repense aux plus anciens, Pierre, Brice, Benjamin, Raphaël et Claire, qui indéniablement ont mis la barre très haute, ainsi que Cyrille, qui continue à m'envoyer des volants très hauts! Je remercie particulièrement Benjamin qui a rythmé nos vies de doctorants. Merci pour ta générosité qui me semble sans limite. Merci Raphaël pour toutes ces discussions techniques que nous avons eu et qui n'ont cessé d'attiser ma curiosité scientifique. Parmi mes "contemporains", je remercie Pauline, Baptiste et Chen avec qui j'ai commencé cette aventure (vous vous souvenez en C VERT?). Merci pour tous ces Tablapizza regrettables avant nos parties de bowling diaboliques. Je vous souhaite également plein de réussite dans les chemins que vous empruntez. J'ai une attention particulière pour ceux qui ont partagé plus récemment leur bureau avec moi. Merci d'abord à Solène et à Wojtek qui ont dû supporter de nombreuses fois ma discrétion. Puis, je tiens à remercier tout particulièrement Florent qui de manière peut-être non soupçonnée m'a véritablement transmis l'esprit de l'informatique. Si aujourd'hui j'ai tant d'attrait pour la *sécu*, le *dev-ops* et dès qu'il s'agit de bidouiller un système (le *hacking* étymologiquement), c'est bel et bien grâce à toi. Il me reste encore beaucoup de challs à valider... J'espère que tu auras encore des *pro tips* à me donner et que l'on pourra 1CLICK au(x) BZHCTF. Merci Angèle pour ta bonne humeur, ton accent québécois et tes néologismes qui formeront bientôt un nouveau dialecte. Merci Alexandre pour tout le sport que tu fais à ma place mais surtout pour cet esprit critique qui te distingue dans le monde de la recherche. Bon courage à vous tous qui ont, ou qui auront, le privilège de passer ces quelques années au sein d'EMSEC.

Plus généralement, je remercie tous les permanents qui nous encadrent, nous transmettent leurs savoirs et leur expérience, qui font vivre leur équipe au quotidien et qui donnent de leur personne pour la faire progresser encore.

J'ai également eu la chance d'encadrer modestement un stage. Merci Nicolas pour tout le travail que tu as accompli et qui m'a énormément servi.

Certains sont tombés dans une marmite de potion magique étant petits, de mon côté il s'agissait d'une caisse de LEGO. Merci à mes parents qui m'ont permis ainsi de développer patience et créativité, des qualités qui me préparaient déjà au monde de la recherche.

Indéniablement, je n'aurais pas pu arriver jusqu'ici sans toi, ma chérie. Merci de m'avoir suivi dans cette aventure, de m'avoir aidé à surmonter les obstacles et d'avoir cru en moi. Je t'aime.

Résumé

Nous avons actuellement un intérêt sans précédent pour les connaissances que nous pouvons extraire des données. La numérisation de la société engendre un flot toujours plus important d'informations, que ses acteurs (entreprises, administrations, associations et même particuliers) tentent de collecter et de valoriser. Cette démarche entreprise sur les données est en réalité très analogue aux industries pétrolière ou minière. Une ressource brute est récoltée puis raffinée pour ne conserver que l'utile. Les termes "exploration" ou "fouille de données" (*data mining* en anglais) prennent alors tout leur sens pour désigner ce domaine de recherche. Concernant l'importance de la "donnée", cette dernière a parfois été qualifiée de "nouvel or noir" ou de "pétrole de demain".

Les objectifs de l'exploration de données sont multiples et varient naturellement selon les secteurs. La vente cherche à profiler ses clients pour de futures recommandations, les acteurs de la santé désirent poser le diagnostique le plus pertinent, la sphère financière souhaiterait prédire le cours des actions, enfin les experts en cyber-sécurité aspirent à se protéger des attaques informatiques de type *zero-day*. Ces objectifs nécessitent d'extraire des informations à partir de données : groupes de clients, règles d'association entre symptômes, tendances du marché boursier ou comportement normal au sein d'un réseau.

En plus de ces activités de fouille de données, plusieurs contraintes s'ajoutent. Par exemple, nous souhaitons répondre aux exigences suivantes: traiter des milliards d'observations, produire des résultats très fiables (sans erreurs) ou être capable d'absorber en continu un débit élevé de données.

Sommes-nous actuellement capables de répondre à tous ces besoins ? Il n'y a clairement pas de réponse simple. Une voiture peut-elle être autonome ? Oui, nous en avons de nombreux exemples. Une voiture autonome peut-elle rouler par mauvais temps sur des routes de campagne sinueuses sans marquage ni signalisation ? La réponse n'est pas aussi évidente. En fait, de nouveaux facteurs remettent constamment en question nos acquis. Certains d'entre eux proviennent de la confrontation avec le contexte réel : les voitures ne circulent pas seulement sur les autoroutes.

Détaillons un autre exemple commun en sécurité informatique. Certaines personnes ont passé un temps conséquent à étiqueter des paquets réseau (bénins ou anormaux) afin d'entraîner de puissants algorithmes capables de détecter des cyber-attaques. Ces algorithmes sont évidemment très efficaces sur les données de test correspondantes, mais que produisent-ils une fois connectés à un réseau totalement inconnu ? Probablement de bien plus modestes résultats. Nous sommes capables de concevoir de puissantes solutions spécifiques, mais l'objectif actuel ne serait-il pas de concevoir une solution générique et adaptable, capable d'apprendre sans information préalable ?

L'émergence de nouvelles exigences est de même un facteur expliquant le défi perpétuel. Aujourd'hui, nous voulons des algorithmes non seulement rapides et précis, mais également interprétables, robustes face à des adversaires voire équitables. Parfois, ces nouveaux besoins peuvent être incompatibles avec les précédents. Un exemple classique est l'interprétabilité des réseaux de neurones. Ces derniers sont simplement des fonctions complexes entraînées à trouver des motifs complexes via des transformations non linéaires. Actuellement, nous aimerions savoir quelle caractéristique (ou combinaison de caractéristiques) est responsable d'un résultat donné, similaire à une rétro-ingénierie de l'algorithme.

Cette thèse a tenté modestement d'aborder quelques-uns des nombreux défis auxquels nous sommes actuellement confrontés en fouille de données (*data mining*), notamment en développant de nouvelles méthodes statistiques, fortes de part leur aspect générique et leur interprétabilité.

Contribution à la détection d'anomalies Mes premières recherches se sont concentrées sur les techniques de détection d'intrusion basées sur la recherche d'anomalies. La littérature dans le domaine est à la fois riche et éparse. Riche des nombreuses contributions existantes mais malheureusement peu disposée à résoudre les problèmes les plus concrets. Le principal problème observable est la nécessité de fixer plusieurs paramètres (ou hyper-paramètres), qui généralement sont difficilement interprétables, résultant en une tâche difficile pour l'utilisateur. Dans le domaine plus général de la détection d'anomalies, un paramètre crucial et largement utilisé demeure le seuil de décision. L'utilisation d'un tel seuil permet généralement d'explorer toutes les capacités de l'algorithme, mais comment le fixer en pratique ? Quelle est sa signification ? Malheureusement, ces questions ne sont presque jamais abordées.

Le problème sur lequel nous nous sommes concentrés était le suivant : à partir de quelques observations, pouvons-nous calculer une frontière (un seuil) qui sépare les anomalies des données normales ? Cela dépend naturellement de la définition d'une

anomalie. En pratique, il s'agit d'un événement inhabituel, avec donc une faible probabilité de se produire. C'est pourquoi nous avons privilégiés les méthodes statistiques pour aborder cette question. Comme principale contrainte, nous désirions développer la méthode la plus générique possible: naturellement non supervisée mais également sans hypothèse de distribution. En gardant à l'esprit le contexte de la cybersécurité, nous souhaitions également qu'un tel algorithme puisse opérer en temps réel, sur des débits de données élevés tout en limitant le nombre d'alertes.

Dans [115] , nous avons développé une nouvel algorithme visant à détecter des anomalies dans des flux de données univariés, nommé SPOT pour *Streaming Peaks-Over-Threshold*. C'est la première fois qu'un détecteur d'anomalies statistiques en temps réel a été conçu sans hypothèse de distribution. SPOT se base sur la théorie des valeurs extrêmes (EVT) qui fournit un modèle générique pour une très grande variété de queues de distribution. Par l'utilisation de la théorie des valeurs extrêmes, il est possible d'estimer un modèle dans les zones à faible probabilité où généralement nous n'avons pas de données. Plus formellement, EVT permet de calculer un quantile z_q tel que la probabilité $q = \mathbb{P}(X > z_q)$ soit aussi petite que souhaitée sans connaître ni supposer la distribution de X.

Dans cette contribution, nous avons adapté les méthodes d'EVT pour calculer et mettre à jour z_q en temps réel. En particulier, ce quantile est directement utilisé pour détecter des évènement anormaux: dès qu'une observation dépasse z_q, elle est déclarée comme anormale car très peu probable (probabilité inférieure à q). Cependant, cette méthode peut être plus largement utilisée pour le calcul automatique de seuils. Si l'on imagine un algorithme complexe produisant des scores d'anormalité bruts, SPOT peut être disposé en aval pour automatiquement apprendre (et mettre à jour) un seuil de décision z_q sur ces scores (avec q fixé au préalable). De manière plus globale, SPOT construit automatiquement un seuil qui lui-même est interprétable: les anomalies sont les évènements de probabilités inférieure à q.

Nous avons également proposé une variante de SPOT, nommée DSPOT pour *Drifting SPOT*. Contrairement à SPOT qui suppose une distribution stationnaire (constante au cours du temps), DSPOT prend en compte la tendance locale et exécute la même tâche que SPOT mais sur les écarts à la tendance. Succinctement, SPOT détecte des pics absolus alors que DSPOT détecte des pics relatifs à la tendance. Enfin nos expériences ont montré que ces algorithmes sont efficaces et qu'ils peuvent être utilisés dans de nombreux domaines tels que la finance, la physique ou la cybersécurité.

Pour compléter ces recherches, nous sommes revenus aux applications en cybersécurité. Dans [114], nous nous sommes concentrés sur l'utilisation de méthodes de *Machine Learning/Data Mining* (ML/DM) pour la détection d'intrusion, pointant notamment les raisons du scepticisme des experts du domaine. Dans cet article, nous présentons également les défis à relever en cybersécurité et nous exposons le rôle que peut jouer (D)SPOT dans ce contexte. Parallèlement à ce travail nous avons développé `netspot`, une sonde réseau (*network intrusion detection system*) détectant des anomalies grâce à SPOT.

Le premier chapitre introduits les méthodes statistiques pour la détection d'anomalies

et promeut leur utilisation. Le chapitre 2 aborde la théorie des valeurs extrêmes et son utilisation pratique alors que le chapitre 3 détaille et expérimente principalement les algorithmes SPOT et DSPOT. Finalement, le chapitre 4 se concentre davantage sur le domaine de la cybersécurité. Nous abordons l'utilisation de méthodes ML/DM pour la détection de cyber-attaques et nous présentons également les propriétés et les atouts de `netspot`.

Contribution au test d'unimodalité L'approche adoptée par (D)SPOT est très générique, mais comme nous l'avons mentionné dans l'introduction, certains défis demeurent. Parmi les nouvelles difficultés, nous pouvons citer la suivante : supposons que nous surveillons un flux de données bimodal (distribution avec deux modes). (D)SPOT peut aisément calculer un seuil supérieur (et inférieur) z_q pour distinguer les événements extrêmes (anomalies). Néanmoins, la zone entre les deux modes est également une région à faible densité mais qui ne peut pas être prise en compte par l'algorithme. Notre objectif initial était donc de détecter ce type de configurations.

Le test d'unimodalité est un sujet relativement confidentiel. La littérature est plutôt parcimonieuse et les principaux travaux ont été réalisés dans les années 1980. Des tests permettant de vérifier si une distribution est unimodale ou multimodale existent mais le problème demeure dans leur adaptation à des flux de données.

Dans [116], nous avons développé un nouveau test appelé *Folding Test of Unimodality* (FTU). Il s'agit du premier test d'unimodalité capable de traiter des flux de données. En plus de cette caractéristique, cette même publication présente également une généralisation en dimension $d > 1$, ce qui fait de la FTU le premier test d'unimodalité capable de traiter des données multivariées. De précédentes publications ont proposé de telles généralisations, mais elles testent unimodalité vs bimodalité, ce qui est un problème plus faible. De plus, ces travaux utilisent des structures de données additionnelles complexes alors que FTU n'est basé que sur de simples statistiques.

Au cours de nos recherches, nous avons également mis en évidence certaines faiblesses de FTU, notamment sa difficultés face aux distributions symétriques. Ces configurations sont susceptibles de se produire dans des espaces de faible dimension où les données ont naturellement moins de degrés de liberté qu'en grande dimension (les symétries sont "plus probables" en un certain sens). Nous avons ensuite proposé le *Hyperplane Folding Test of Unimodality* (HFTU) qui est basé sur le même mécanisme de "repliage" (*folding mechanism*) que FTU mais avec une différente généralisation multivariée. HFTU résout certaines faiblesses de FTU mais sa principale force est qu'il construit un hyperplan disposant de nombreuses propriétés.

Comme application directe du test d'unimodalité, nous avons conçu un algorithme de *clustering* basé sur l'hypothèse d'unimodalité des clusters (UAC) qui indique qu'un cluster est valide dès lors qu'il est unimodal. En fait, l'UAC est beaucoup plus générale que l'hypothèse gaussienne très commune des clusters. Malheureusement, avant FTU et HFTU, nous n'avions aucune méthode efficace de *clustering* sous UAC. Ainsi, nous avons développé $\Phi-$means, un algorithme basé sur $k-$means qui trouve progressivement le "bon" k (le nombre de clusters) grâce à HFTU. Cette dernière partie de notre recherche

est naturellement la moins aboutie mais nos expériences montrent déjà que $\Phi-$means est comparable à l'état de l'art.

Le chapitre 5 présente la littérature sur les tests d'unimodalité et souligne le besoin de développer un nouveau test. Le chapitre 6 présente le concept de base du mécanisme de "repliage". Cette idée sera fondamentale pour développer à la fois FTU (chapitre 7) et HFTU (chapitre 8). Enfin, le dernier chapitre adapte HFTU pour concevoir un nouvel algorithme de *clustering* ($\Phi-$means).

Implémentations Durant cette thèse, je me suis efforcé à rendre notre recherche la plus concrète et la plus utile possible. Ainsi, j'ai accordé un temps conséquent dans la documentation et la diffusion de mes implémentations. Tout le code issu de nos projets est notamment disponible sur Github: `https://github.com/asiffer`.

Sans aucun doute que de simples scripts sont suffisants pour expérimenter une idée. Cependant, en plus de la volonté de fournir du code performant, je souhaitais aller plus loin en poursuivant ces deux objectifs :

- facilité d'accès (open-source, paquets `debian`, paquets `python3`, images `docker` ou paquet `snap`)
- simplicité d'utilisation (interfaces haut-niveau, documentation, sites internet)

Contents

Introduction

We currently have an unprecedented interest for all the knowledge we can extract from data. Indeed, the digitization of the society drowns its actors under information. Even if our data could have poor concern not long ago, now we definitely want to value them. Purposes are naturally different. Supermarket or internet retailers want to know their customers to offer them relevant [sometimes so tempting] next item to buy. Health stakeholders want to make the right diagnose based on patterns extracted from previous patients. They are also likely to know if some benign symptoms warn more serious diseases. In the financial field, one would like to predict stock prices and also detect when an unusual or unexpected event occurs. Cyber-security experts want first to detect abnormal behaviors on their systems with the ultimate goal of protecting against zero-day attacks. These tasks require to extract information from data: clusters of customers, association rules between symptoms, trends of the stock market or normal behavior within a network.

In addition to these data mining tasks, several demands constraint them. For instances, we would like some of the following abilities: treat billions of records, output very *reliable* results (no errors) or deal with high throughput streaming data. Scientists have clearly worked hard to find ways to solve the aforementioned problems under these constraints.

Are these problems solved? Undoubtedly, there is no simple answer. Can a car be autonomous? Yes, it can and we have many examples of that. Can an autonomous car run under bad weather on country side winding roads without markings or signs? The answer is not as obvious. Actually, new factors constantly challenge what we already

can do. Some of them come from the confrontation with real-world contexts: cars are not only running on highways, but let us give another example coming from the cyber-security field. Some people have spent much time to label network packets (benign or anomalous) so as to train powerful algorithms now able to flag cyber-attacks. Of course, these algorithms are very efficient on the related test set but what does this algorithm output if it is plugged on a different network? Probably poor results. We are able to build powerful network anomaly detectors but the current challenge is to design a generic solution able to *cold* learn without labels.

The emergence of new demands is also a factor explaining the perpetual challenge. Now, we do not want only fast and accurate algorithms but we are also looking at interpretability, robustness against adversaries and even fairness. Sometimes these new demands might totally go against previous ones, at least within the framework commonly established. A classical example is the interpretability of neural networks. The latter are merely complex functions trained to find complex patterns through non-linear transformations of the features. Now we would like to know which feature (or combination of features) is responsible of a given result, like a reverse-engineering task.

This thesis has tried modestly to tackle few of the numerous challenges we are currently facing in data mining, particularly by developing new statistical methods. Obviously, we may wonder why this specific class of methods, nonetheless they are more generic than we could think. As an illustration, let us look at a famous meme we may have recently encountered on the Internet (we reproduce it on figure 0). It mainly criticizes the fact that research field names have emerged (like "machine learning") but without tackling new challenges.

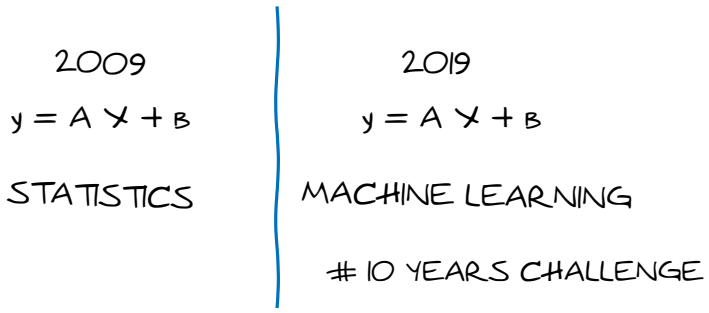


Figure 0: *Recent witty meme comparing "old" statistics with the "recent" machine learning field. In a word, the research domain name has changed but problems are the same*

This picture is rather wrong but obviously there is a little bit of truth in it. Machine learning theory is built upon statistics and naturally, many algorithms we use are actually based on statistics, even if the latter could be hidden. Here, we are not clearly speaking about data mining, but the latter may use several similar techniques as machine learning does. For instance we may see regression as a simple least-square problem. However, regression also aims to estimate the conditional expectation $\mathbb{E}(Y|X)$ where Y is the

target and X the predictor. One can consider principal component analysis (PCA) as a best subspace projection in terms of euclidean distance but this is also a procedure which optimizes the variance of the projected data. Finally, clustering can be viewed as both a distance-based problem and a search for high density regions (also known as *modes*). Thus, statistics remain fundamental and statistical techniques are likely to have a wider impact than other methods.

Another key asset of statistics is their interpretability. It is easier to understand an anomaly detector saying "there's a 99% chance that this observation is an anomaly" than another one saying "the anomaly score of this observation is 2019". Obviously, explain the result does not mean explain their causes, but this is clearly a first step towards a general understanding of an algorithm.

This thesis is divided into two parts where we first present our contributions to anomaly detection and then those to unimodality testing.

Contribution to anomaly detection Immersed in a cyber-security company, my initial researches were focused on anomaly-based techniques for intrusion detection. As we will see in the end for the first part, the literature is both rich and poor. Rich of the numerous existing contributions, poor in the willingness to solve real-world issues. The main issue we may observe is the need to set several parameters or hyper-parameters, which generally lack meaning. So, how should the user set them? In the more general field of anomaly detection, one crucial and very often used parameter is the final decision threshold. Of course, it very convenient to plot ROC curves and compute AUC but how to set it in practice? And what does this threshold really means? These issues are almost never tackled.

The problem we focus on became the following: given some data, can we compute a boundary which separates anomalies from normal data? It naturally depends of what is an *anomaly*. In practice, it is an unusual event, so with a low probability to occur. Therefore we naturally focus on statistical techniques to solve such a problem. Moreover, we also have some constraints to develop a method as generic as possible: data are not labeled (unsupervised) and their distribution is unknown (no distribution assumption). Keeping in mind the cyber-security context, we also aimed to treat high throughput streaming data: it means that such an algorithm should be able to work on streams while limiting the number of alerts.

In [115], we have developed a new algorithm aimed to find outliers in univariate streaming data, called SPOT for Streaming Peaks-Over-Threshold. It is a first time a statistical streaming anomaly detector has been designed without distribution assumption. SPOT is based on Extreme Value Theory (EVT), a framework which gives a model to a very wide variety of distribution tails. By using EVT, we are able to estimate a model in low probability regions, where we generally have no data. More formally, EVT allows to compute a quantile z_q such that $q = \mathbb{P}(X > z_q)$ is as low as desired without knowing the distribution of X.

In this publication, we have adapted methods from EVT to compute and update z_q over streaming data. In particular this quantile is directly used to perform anomaly

detection: once an observation is greater than z_q, it is declared as abnormal (because it has a very low probability, i.e. lower than q). However, such a technique can be more widely used to perform automatic thresholding. Let us imagine a complex algorithm producing raw anomaly scores, SPOT can be put as back-end so as to automatically learn (and update) a decision threshold z_q on the scores, given the value of q. More generally, SPOT automatically builds a threshold that is itself interpretable (anomalies are events with probability lower than q).

In addition to SPOT, we also proposed a variant, called DSPOT for Drifting SPOT. Unlike SPOT which assumes a stationary distribution, DSPOT can take into account the local trend and then perform the same task as SPOT but over drifting streams. In a word SPOT flags absolute peaks while DSPOT flags relative ones. Finally our experiments show that these algorithms are efficient and can be used in several different contexts like finance, physics or cyber-security.

To complete this research circle, we eventually returned to the cyber-security application. In [114], we focus on the use of ML/DM techniques for intrusion detection, pointing out the reasons why experts are skeptical. Besides, we also introduce the challenges to address and we detail how (D)SPOT can be used to solve some of them. In a background thread, we developed `netspot`, an anomaly-based Intrusion Detection System (IDS) powered by SPOT. Despite its simple design, `netspot` has been tested on real-world data and has shown its ability to detect cyber-attacks.

The first chapter introduces the statistical methods for anomaly detection and advocate the broad use of this class of methods. The chapter 2 deals with Extreme Value Theory (EVT) and a way to use it in practice, while the chapter 3 mainly details the SPOT and DSPOT algorithms. Finally the chapter 4 concentrates on the cyber-security application. We discuss about the attempts to use ML/DM methods to detect cyber-attacks and we present the design, the power and the efficiency of `netspot`.

Contribution to unimodality testing The approach taken to develop (D)SPOT is very generic but, as mentioned during the introduction, some challenges remain. One would be the following: let us assume that we monitor a bimodal stream. (D)SPOT can easily compute upper and lower thresholds z_q to discriminate extreme events (anomalies). Nonetheless, the area between the two modes is also a low-density region but that cannot be monitored by the algorithm. So our goal was initially to detect this kind of configurations.

Unimodality testing is clearly not the most trendy topic. The literature is rather sparse and the main works have been done in the 80s. Does it exist a statistical test to check whether a distribution is unimodal or multimodal? Yes, but the burning issue is that they cannot be easily adapted to streaming data.

In [116], we have developed such a new test called Folding Test of Unimodality (FTU). This is the first unimodality test able to deal with streaming data. In addition to this feature, this same publication also presents a generalization to higher dimensions, making FTU also the first test of unimodality able to tackle multivariate data. Some previous

attempts had been proposed in the 80s but they test unimodality versus bimodality, which is a weaker task. Moreover, these works use complex data structures while FTU is only based on simple statistics.

During our research, we also have pointed out some weaknesses of FTU: it particularly suffers from distribution symmetries. These configurations are likely to occur in low-dimensional spaces where data have naturally fewer degrees of freedom than in high-dimensional spaces (symmetries are "more probable" in some ways). Alongside FTU, we then proposed the Hyperplane Folding Test of Unimodality (HFTU) which is based on the same univariate folding mechanism as FTU but with another multivariate generalization. HFTU solves some weaknesses of FTU but it especially builds an hyperplane which has very convenient properties.

As a direct application of unimodality testing, we have designed a clustering algorithm based on the *unimodality assumption of the clusters* (UAC) which states a cluster as valid once it is unimodal. Actually, UAC is far weaker than the very common gaussian assumption of the clusters because it catches a wider class of cluster shapes. Unfortunately, before FTU and HFTU, we had no efficient method to cluster under UAC. So that, we have developed $\Phi-$means, a wrapper around $k-$means which incrementally finds the right k thanks to HFTU. This last part of our research is naturally the least accomplished but our experiments already show that $\Phi-$means is comparable to the state-of-the-art.

The chapter 5 presents the literature about unimodality testing and highlights the reason to develop a new test. The chapter 6 presents the core concept of the folding mechanism. This idea will be fundamental to develop both FTU (chapter 7) and HFTU (chapter 8). Finally, the last chapter adapts HFTU to design a new clustering algorithm ($\Phi-$means).

Publications

Conference/Journal	Article
C&ESAR 2018	*Intelligent Thresholding* [114] (A. Siffer)
KDD 2018	*Are your data gathered?* [116] (A. Siffer, P. A. Fouque, A. Termier, and C. Largouët)
KDD 2017	*Anomaly detection in Streams with Extreme Value Theory* [115] (A. Siffer, P. A. Fouque, A. Termier, and C. Largouët)

Implementations During this thesis, I have put a great deal of effort into making our research as practical as possible. Thus, I spent much time to distribute our work. All our code is available on my Github repository:

`https://github.com/asiffer`

We may understand that providing scripts is sufficient to show an idea. Actually, I have tried to go further by pursuing these two objectives:

- *Easy to get* (open source, `debian` packages, `python3` packages, `docker` images, `snap` packages)
- *Easy to use* (library bindings, documentation, web pages)

The table 0 gives the list of the projects we made available (with GPLv3 license[1]). Besides, every chapter will present its related implementations. Some of them may not currently be publicly available because of blind submission constraints.

Projects	Details
`libspot`	`C++` library which implements the SPOT and DSPOT algorithms
`python3-libspot`	`python3` bindings to `libspot`
`libfolding`	`C++` library which implements the folding test of unimodality (FTU)
`python3-libfolding`	`python3` bindings to `libfolding`
`Rfolding`	`R` package to perform FTU
`hftu`	`python3` package which implements the Hyperplane Folding Test of Unimodality (HFTU)
`wtvmeans`	`python3` library which implements several $k-$means wrappers including $\Phi-$means
`netspot`	Intrusion Detection System (IDS) powered by SPOT
`gopula`	`Go` package to deal with archimedean copulas

Table 0: *List of projects developed during this thesis and available on Github*

[1] `https://www.gnu.org/licenses/gpl-3.0.en.html`

Part I
Anomaly Detection

According to Chandola, Banerjee, and Kumar, "anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior" [16]. Anomaly detection has attracted considerable attention due to its importance in many real-world applications including intrusion detection, fraud detection, energy management, health care, insurance and finance.

Indeed, finding anomalies, also known as *outliers*, is a relevant task insofar as it brings paramount information in the operational or business perspective. For example, finding anomalous IP packets can lead to detect local network intruders. Quickly detecting anomalies in stock prices allows to initiate buy or sell orders. An anomalous output of an industrial sensor could mean a breakdown on a machine. Finally an abnormal transaction can reveal some frauds and even perpetrators.

Many techniques, generic or specific, have been developed over time to detect outliers. One major challenge is generally the lack of labeled data, meaning that we have not a clear idea about what is a normal or abnormal observation. Especially, getting anomalous examples is often impossible because such events have not occurred yet. Even if some observations are available, they may be insufficient to learn what is an anomaly. Can we have access to several spacecraft sensors before their crash? Can we have access to several weather measurements preceding a century tide? In some contexts, the abnormal behavior can also change over time, deprecating then the initial labels. In all other cases, getting representative labeled data is "often prohibitively expensive" [16].

Because of this labeled data starvation, unsupervised techniques are undoubtedly the most suitable. They commonly rely on the following assumption: normal instances are far more frequent than anomalies. Thus they commonly model what is normality, looking for deviant behaviors (anomalies).

Several categories of anomaly detection methods exist. The vast majority of them actually produce a score (i.e. a real value) estimating the degree of normality or abnormality of an observation. The main weakness of anomaly detection remains the thresholding stage: below/above which threshold is the observation declared abnormal? Whereas a great amount of intelligence is naturally embedded in the scoring stage, this latter decision should not be neglected. Unfortunately, this task is almost always let to the user without further help. Here two interrelated issues arise in practice: (1) how to set the decision threshold? (2) What does the threshold value mean?

In this part, we will focus on statistical methods for anomaly detection. To our point of view, these methods have a role to play in every anomaly detector, particularly during the thresholding stage. However, the most efficient statistical methods generally rely on strong assumptions (e.g. data are gaussian distributed). Our first contribution is to propose SPOT, a new anomaly detection algorithm based on Extreme Value Theory (EVT). EVT is a powerful statistical framework which allows to accurately compute extreme quantiles without distribution assumption. We have adapted some EVT results to compute and update such quantiles online. By using these quantiles as thresholds, SPOT is able to detect anomalies over high throughput univariate streaming data. A variant called DSPOT has also been developed to deal with drifting streams.

Finally, we will investigate the cyber-security field which has a real need for anomaly

detection. We will see that a great amount of work have been done to build anomaly-based intrusion detection systems (IDS) but generally by putting aside practical aspects. However, some new real-world solutions also emerge in the literature and we will analyze some of them, pointing out their pros and cons. In the end, we will detail `netspot`, a new anomaly-based IDS we have developed powered by the SPOT algorithm. In particular, we will see that the simplicity of our design is enough powerful to detect real-world attacks.

1

Statistical methods for anomaly detection

In the statistical point of view, "an anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed" [5]. In practice, statistical techniques to detect anomalies rely on the following assumption: "Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model" [16].

Labels are then generally not needed since the normal behavior is defined by the underlying model. Thus, while we generally distinguish supervised algorithms from unsupervised ones, statistical techniques are rather put into two categories: parametric or non-parametric. Parametric techniques assume a model (i.e. a parametric distribution) although non-parametric techniques find a model from input data.

1.1 Parametric techniques

As explained beforehand, parametric techniques assume that normal observations are generated by a parametric density f_θ where θ is a parameter to estimate from data. Once such a model is estimated, we can easily compute quantiles and then discriminate anomalies from normal data. Another approach is the statistical hypothesis test. Given an observation x, the null hypothesis is defined as H_0: "x has been generated from f_θ". Within this framework, x is declared as an outlier when the test rejects H_0.

From these two methods, scores can also be derived. In the former case, $1/f_\theta(x)$ is a classical way to quantify an observation (the higher, the more unlikely as $f_\theta(x)$ is the likelihood to observe x). In the latter case, the corresponding test statistic can naturally be used.

Gaussian-based models The gaussian distribution is widely used to fit data and it is a reasonable assumption in many cases. Its parameters (mean μ and variance σ^2) can easily be computed through maximum likelihood.

The first basic outlier technique is the so-called *three-sigma rule*: an observation is declared as abnormal when it lies out of the $\mu \pm 3\sigma$ region. For the normal distribution, this area embeds about 97.3% of the probability mass (see figure 1.1).

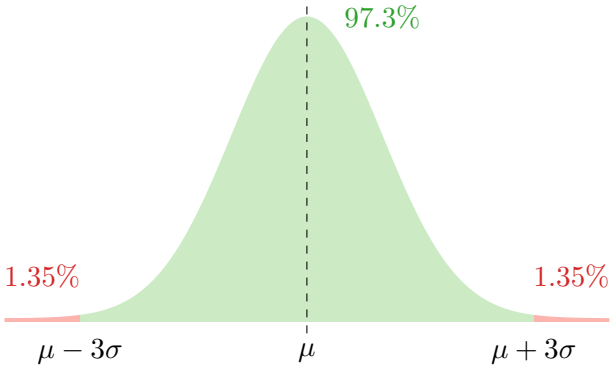


Figure 1.1: *The 3σ rule on gaussian distribution*

This rule can be extended in two ways. First we can actually take another quantile keeping the gaussian assumption (see [1] for a recent example). Otherwise, this rule may be used in a non-parametric way (not assuming any distribution). It commonly emerges when thresholds on $z-$scores are used to discriminate anomalies [22, 96]: the $z-$score is defined by $z = |X - \mu(X)| / \sigma(X)$ (it corresponds to the distance to the mean and relative to the standard deviation) so $z \leq k$ is equivalent to $\mu(X) - k\,\sigma(X) \leq X \leq \mu(X) + k\,\sigma(X)$.

A related rule is the *box-plot rule*: normal data lie in the region $[Q_1 - 1.5\Delta, Q_3 + 1.5\Delta]$ where Q_1 and Q_3 are the first and third quartile and $\Delta = Q_3 - Q_1$ is the inter-quartile range. For instance this technique is a component of a more complex detection algorithm in [108]. The box-plot rule is equivalent to the three-sigma rule for the gaussian distribution.

The Grubbs' test [45] is a statistical test based on the law of the $z-$scores when normal instances are gaussian distributed. Instead of thresholding the $z-$scores, it works with the law of the Grubbs' statistic. Given a sample $X_1, X_2, \dots X_n$, it is defined by:

$$G_n = \frac{1}{s_n} \max_{1 \leq i \leq n} |X_i - \mu_n| ,$$

where μ_n and s_n are the sample mean and sample standard deviation. G_n is then the the largest deviation from the mean (in standard deviation units). The distribution of G_n can be estimated through Monte-Carlo simulations (approximated formulas based on Student's quantiles are also used). In practice we have to assume getting $n-1$ normal observations $X_1, \dots X_{n-1}$, then, given a new observation X_{new}, the value $G_n(X_{\text{new}}) = |X_{\text{new}} - \mu_n| / s_n$ is computed and compared to a quantile $q_{\alpha,n}$. In [39], the Grubbs' test is performed on neighbors distances to detect abnormal printing techniques.

Rosner [103] proposed an extension of the Grubbs' test, dealing with multiple outliers, called Generalized ESD (Extreme Studentized Deviate) test. The idea is to incrementally remove the observations x_i that maximize $|x_i - \mu|$. At each deletion, the Grubbs' statistic is computed. Assuming at most k outliers, k values are removed and we get k Grubbs statistics $G^{(1)}, \dots G^{(k)}$. The number of outliers is determined by finding the largest $i \in \{1, \dots k\}$ such that $G^{(i)} > q_{\alpha,n-i}$. This approach has notably been used in [54] to find anomalies in "cloud" metrics (systems, applications and business metrics).

Several multivariate variants of the Grubbs' test have also been proposed. In [75], the Grubbs' test is performed on the mahalanobis distance $D^2 = (X - \mu_n)^T \Sigma_n^{-1} (X - \mu_n)$ where X is a multivariate observation, μ_n the sample mean and Σ_n the sample covariance matrix (it also assumes that X follows a multivariate gaussian distribution).

Regression models Another class of parametric techniques for anomaly detection is regression model. The principle is to fit the data to a given model and consider the residuals as anomaly scores. ARIMA (Autoregressive Integrated Moving Average) models are among the most popular. In [102], an ARIMA model is used to detect anomalies in wireless sensors network traffic. A instance is declared as abnormal when the absolute relative error with the prediction is higher than a given threshold. However, confidence bounds of the ARIMA model are used in [70, 47] to detect the anomalies. In [70], 95% bounds are used to detect theft in power grids while 99.5% bounds are computed in [47] to detect outages on Internet traffic. Learning the model is likely to be the main hardship especially if it contains abnormal data. For that purpose, some models, called *robust regression*, have been developed [105].

Mixture models Finally, mixture models aim to use a mixture of distributions to model the data. One may either assume a distribution for normal data and another one for anomalies or only model the normality with such a mixture (anomalies being in low probability regions). In [30], data are assumed to be generated by the distribution $\mathbf{D} = (1 - \lambda)\,\mathbf{M} + \lambda\,\mathbf{A}$, where $\mathbf{M}$ is the distribution of normal data and $\mathbf{A}$ those of anomalies. The parameter λ corresponds then to the proportion of abnormal observations. An

observation x is treated as follows: a priori x is considered as normal (a log-likelihood is computed), then the log-likelihood is recomputed assuming $x \sim \mathbf{A}$. If the log-likelihood increases by more a given threshold, x is considered as an anomaly.

As examples of the other strategy, the contributions [76, 107] model normality with gaussian mixtures to detect anomalies in crowded scenes. The specificity in [76] is the use of a dynamical model. Once again a threshold is set on the density to discriminate low probability regions.

1.2 Non-parametric techniques

Non-parametric techniques do not assume a model for the input data. Their aim is indeed to find what is the distribution of the observations.

Histograms The first natural way to estimate a density is to build an histogram of the observations. Histograms are particular used in intrusion detection because they are scalable and well adapted to the streaming context [42, 66].

The basic use of the histogram is to flag observations which do not fall into bins. However many hardships emerge like the choice of the size of the bins which tune both the false positive and false negative rates. In [42], the Histogram-based Outlier Score (HBOS) is defined by

$$\mathrm{HBOS}(x) = -\sum_{i=1}^{d} \log\left(\mathrm{hist}_i(x)\right),$$

where the hist_i are the dimension-wise histograms. Besides the underlying anomaly detector may use a procedure to recompute the bins sizes. In [65], authors use a truncated histogram called *iceberg-style* histogram built with a frequent items search algorithm [63]. This histogram only keeps bins with frequency higher than a given threshold.

A smarter use of the histograms consists in comparing profiles. A common way is to use a distance between these histograms. In [66], authors use the Mahalanobis distance to estimate the histograms similarity. Then a clustering stage is performed so as to build normality profiles. In [56], the Kullback-Leibler divergence is privileged to compare histograms. A user-defined threshold is finally used to flag anomalies.

Kernel methods Given n observations $x_1, x_2, \dots x_n$, a kernel estimator of a univariate density f is commonly given by

$$\hat{f}(x) = \frac{1}{n\,h} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where K is the kernel and h a smoothing parameter. Kernel methods are actually a generalization of histograms: a kernel estimator using a characteristic function (constant function on a bounded support) as a kernel function is an histogram. Such an estimator can also be generalized to multivariate distributions. In [94], the output of an auto-encoder is modeled by a kernel density estimate, with gaussian kernel. A threshold is

used on this density to detect anomalies. Authors of [111] proposed an outlier detector inspired by Local Outlier Factor (LOF) [12]. They use kernel estimates to model and then compare local densities (gaussian and epanechnikov kernels are suggested).

Comparison with parametric techniques Actually there are pros and cons in using either parametric or non-parametric methods. As seen above, parametric techniques need to set a model for the data. Thus, they commonly require strong distribution assumptions which may not be adapted in some contexts. However, as long as the model is valid they are efficient and have a real ability to generalize. Non-parametric techniques make less assumptions since their model is based on the observations (*data-driven*): they are then more general and adaptable. Nonetheless, they bring few information in the region far from the training observations (hard to generalize elsewhere) and they often need extra parameters (like bins width or kernel bandwidth) which are not easy to set in the general case while their impact is quite high.

1.3 Why using statistical methods?

In this first chapter, we only deal with statistical methods to anomaly detection. Obviously, several other families exist: classification based, clustering based, nearest neighbor based, information theoretic based or spectral based... The survey of Chandola *et al.* [16] remains a reference for anomaly detection. Here we explain why a particular interest must be put on statistical methods.

Anomaly detector design The common point of the vast majority of anomaly detectors is that they rely on scores. Every algorithm computes a normality or abnormality score s_i for all the observations x_i. But the whole anomaly detection process is made in two parts: *scoring* and *thresholding* (see figure 1.2). The scoring part is generally the part where the intelligence lies: all the efforts are made to build a score which really describes the "outlierness" of an observation. Then a threshold z is often used in back-end to make the final decision (thresholding part): is $s_i > z$ then x_i is flagged as an anomaly otherwise it is considered as normal (we assume s_i to be an abnormality score).

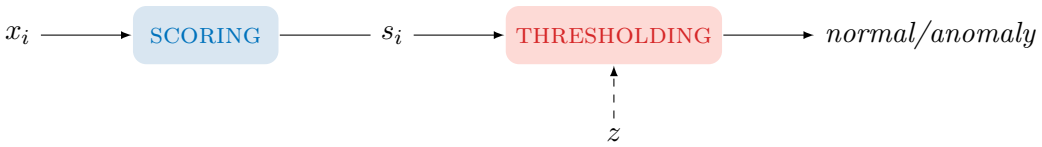


Figure 1.2: *Generic structure of anomaly detectors*

The parameter z can often be seen as a detection/false-positive regulator: a "high" value for z will catch only the most abnormal data, then many other real anomalies will be missed (low detection rate) but the probability to be wrong will be reduced (low false-positive rate). The converse phenomenon happens when z is "low". The thresholding

stage can also be made by a top$-k$ algorithm returning the k most abnormal observations (here the parameter k takes the same role as z) like in [25].

Thresholding issue Actually, there is very little interest in the thresholding stage. In several works, only the scoring part is even presented: the ROC curves and AUC are used to assess the performance of the anomaly detector (see [41, 83, 128]). When the whole detector is presented, decision thresholds are usually set after a fine-tuning step to get the best results on some datasets but either no formal procedure exists for setting the thresholds or it may require some labels to set it in practice [58].

This gap may lead to an obvious practical issue: given some data and a scoring algorithm, how to set the decision threshold? Such a problem particularly arises when the context changes (other data sources, concept drift etc.). Another problem is the lack of interpretability: scores have generally no meaning (except that they smartly rank the observations) so the final threshold-based decision is likely to be hard to explain.

Below, we give some arguments explaining why statistical methods should be used in anomaly detection, and particularly in the thresholding stage.

- Statistical methods, even designed to be standalone anomaly detectors, can be used in the thresholding part to automatically set decision thresholds. As a basic example, the scores can be assumed normally distributed and then we may use $\mu + 3\sigma$ as a decision threshold, but more complex methods such as those described above can be adapted to perform this task.

- Statistical methods are adapted to unsupervised scoring algorithms since they can even learn a relevant decision threshold when the dataset has some outliers.

- While hard-set threshold may lack of meaning, statistical methods bring a little amount of interpretability within the whole pipeline: commonly an instance will be declared as abnormal because its probability to occur is less than p. The probability p is then the new threshold but it is also the false-positive rate. So, with this piece of information, it is far easier to set it regardless of context or data.

- Statistical methods are generally fast so quite relevant to build streaming anomaly detectors. Indeed, they can easily be adapted to dynamic context where static thresholds are helpless.

As recent examples tackling the thresholding issue we may cite [58] which uses a dynamic threshold computed from the mean and the variance of the scores. In [95], authors rather use a concentration inequality to make the final decision (Cantelli's inequality). It especially implies a bound on the false-positive rate. Finally, the contribution [85] notices that several metrics over the Internet follow the Benford's law. Under this assumption it uses the Kolomogorov-Smirnov test to detect outliers.

Concluding remarks Here we have presented the main statistical techniques to detect outliers. Despite the simplicity of some of them, they remain efficient and they are still widely used. Moreover we advocate their need in the thresholding stage that most anomaly detectors perform. Unfortunately, as research does not focus on this part (rather on the scoring) some practical shortcomings arise when anomaly detection algorithms are used in practice (parameter settings). We think that statistical methods are relevant to fill this gap in building automatically a smart threshold in back-end of a possibly more complex scoring algorithm.

In the next section we will present the Extreme Value Theory (EVT), a powerful statistical framework to estimate distribution tails. This latter will be used to build a new statistical anomaly detector able to deal with high throughput streaming data.

2

Extreme Value Theory

In 1953, dikes were not big enough to turn the waves down during a storm in the Netherlands. In the aftermath, a committee is requested to study the phenomenon and evaluate dikes' height. In this context, the tide heights are modeled as the realizations of independent and identically distributed (iid) random variables. So two questions raise :

- Let $q \in (0,1)$, can we calculate z_q (the dikes' height to build) such that the probability to exceed z_q is lower than q ?
- Let z_q be a threshold, can we calculate the probability q to exceed this value (the flooding risk) ?

Many techniques allow the scientist to answer both questions. First we can model the phenomenon, find the law of interest and then calculate all the desired quantiles and probabilities. An other method would be to estimate them empirically. The limits of these methods are the following : data are not devoted to be standard distributed (gaussian, uniform, exponential etc.) so the model step could be not suitable. Moreover, if we want to predict *extreme* events (as tidal waves), like rare or unprecedented events, the empirical method will not give accurate estimation (an unprecedented event would have a probability equal to zero).

Finally, a third way is likely to be more robust against these drawbacks : the *extreme value theory*. Some powerful results show that the maxima of sequences of iid variables converge in distribution to only a restricted family of probabilistic laws (apart from the initial law). Thus it is possible to estimate some extreme events without knowing clearly the law of these events in the common case.

Actually, we are facing these problems in many fields : insurance, climatology, energy, seismology and cyber-security. In the latter case, intrusion detection is typically a task requesting to build such *extreme thresholds*: the amount of incoming data is so huge that q must be very low to limit the numbers of flagged events.

PROBLEM 2.1 *Let X be a measure of interest and $X_1, \dots X_n$, n independent observations of this measure. For q as small as desired, set the lowest threshold z_q such that the probability of $X > z_q$ is lower than q.*

In this chapter, we introduce EVT and how it could be used to compute extreme quantiles z_q. The first sections mainly present general results which can be found in the rich references made by Coles [19] and Beirlant *et al.* [8]. They lead to the section 2.5 which deals with our method we use in practice to compute the desired quantiles.

⚠ This chapter tries to explain how EVT is developed until the practical computation of extreme quantiles. It exposes technical mathematical results which are not mandatory to understand the next chapters.

2.1 Law of extreme events

Here we develop the model at the basis of extreme value theory. EVT particulary aims to analyze extreme events by studying the random variable

$$M_n = \max\{X_1, X_2, \dots, X_n\},$$

where $X_1, X_2 \dots, X_n$ are n iid random variables with common cumulative distribution function (cdf) F. In a word, the most extreme event (the maximum) of a batch of observations is considered.

We will note $\bar{F} = 1 - F$ their tail distribution function (also called *survival function*). First of all, the distribution of M_n can be derived from F:

$$\mathbb{P}(M_n < x) = \mathbb{P}\left(\max_{1 \le i \le n}(X_i) < x\right) \underset{\text{independent}}{=} \prod_{i=1}^{n} \mathbb{P}(X_i < x) \underset{\substack{\text{identically} \\ \text{distributed}}}{=} \prod_{i=1}^{n} F(x) = F^n(x)$$

Unfortunately, this result does not help directly since F is unknown. Thus, we rather try to directly model F^n, by looking at its behavior when $n \to \infty$. It means that the most extreme event is studied asymptotically, with a huge amount of observations.

If we note $\tau = \sup\{ x \mid F(x) < 1 \} \in \mathbb{R} \cup \{+\infty\}$, we get $\lim_{n \to +\infty} \mathbb{P}(M_n < x) = 0$ when $x < \tau$ and $\lim_{n \to +\infty} \mathbb{P}(M_n < x) = 1$ when $x \ge \tau$. The limit being trivial M_n

cannot be studied in this way that is why we have to theoretically ensure that there exists two sequences $(a_n)_{n\geq 1}$ with $\forall n \geq 1, a_n > 0$ and $(b_n)_{n\geq 1}$ such that :

$$\mathbb{P}\left(\frac{M_n - b_n}{a_n} < x\right) = F^n(a_n x + b_n) \underset{n\to+\infty}{\longrightarrow} G(x) \tag{2.1}$$

These two sequences play a normalization role. Intuitively, they tune the window where M_n is studied with respect to n: the maximum of few observations (low n) is unlikely to be in the same area than the maximum of a huge number of observations (high n) because the underlying distribution F is naturally more explored in the latter case. Practically, this condition is not so restrictive. For instance, the main part of the standard distributions respect it.

EXAMPLE 2.1 (Exponential distribution) *Let $\lambda > 0$ and let us take $F(x) = 1 - \lambda e^{-\lambda x}$. Let us introduce two sequences $(a_n)_{n\geq 1}$ and $(b_n)_{n\geq 1}$, for all $n \in \mathbb{N}^*$ we get*

$$F^n(a_n x + b_n) = \left(1 - \lambda e^{-\lambda(a_n x + b_n)}\right)^n$$

If we assume $\forall n \in \mathbb{N}^*, a_n = \lambda^{-1}, b_n = \lambda^{-1}\log(\lambda^{-1} n)$, then

$$F^n(a_n x + b_n) = \left(1 - \frac{1}{n}e^{-x}\right)^n \underset{n\to+\infty}{\longrightarrow} G(x) = \exp\left(-e^{-x}\right).$$

Therefore G is the limit cdf of $\lambda M_n - \log(n/\lambda)$, where M_n is the maximum of n independent exponential random variables (with parameter λ).

The Fisher-Tippett-Gnedenko theorem [40, 35] presented below is the fundamental theorem of EVT. It shows that, under the presented condition, the empirical maximum can follow only a narrowed family of laws. In a more illustrated way, this theorem claims that the tails of all the densities are roughly the same (it will be detailed in the next paragraph). For all of us mathematicians, this result for the maximum might be overwhelming as it is really analogous to those of the central limit theorem for the mean.

THEOREM 2.1 (Fisher-Tippett-Gnedenko) *Let $(X_i)_{i\geq 1}$ a sequence of iid random variables. If the condition (2.1) is verified then G belongs to the generalized extreme value (GEV) distributions family, defined by:*

$$G_{\gamma,\mu,\sigma} : x \mapsto \exp\left(-\left(1 + \gamma\left(\frac{x-\mu}{\sigma}\right)\right)^{-\frac{1}{\gamma}}\right), \quad \gamma, \mu \in \mathbb{R}, \sigma > 0 \quad s.t. \quad 1 + \gamma\left(\frac{x-\mu}{\sigma}\right) > 0$$

By continuity, G_0 is defined as $G_0 : x \mapsto \exp\left(e^{-x}\right)$.

REMARK 2.1 *In the next parts, we will say that $F \in \mathcal{D}_\gamma$ when $\mu \in \mathbb{R}, \sigma > 0$ and two sequences a, b exist such that $F^n(a_n x + b_n) \to G_{\gamma,\mu,\sigma}(x)$.*

In fact, the distribution of M_n is not directly known, but those of $(M_n - a_n)/b_n$ (like in the example 2.1). But when n is large

$$\mathbb{P}(M_n < x) \simeq G_{\gamma,\mu,\sigma}\left(\frac{x - a_n}{b_n}\right) = G_{\gamma,\mu',\sigma'}(x),$$

so M_n follows also a GEV distribution. Finally, finding the *extreme value index* γ is paramount to study the behavior of M_n, unfortunately it remains very challenging.

Tail behavior The GEV family can be divided into three parts according to γ (three *attraction domains*): $\gamma > 0$ (Fréchet domain), $\gamma = 0$ (Gumbel domain) and $\gamma < 0$ (Weibull domain). The domain of γ characterizes the limit behavior of the maximum, so it strongly depends on F, as it is described in table 2.1 and illustrated on figure 2.1. Briefly, the shape of the tail of F is closely related to the limit behavior of the maximum.

Domain	Tail behavior ($x \to \tau$)	Distribution example
$\gamma > 0$	Heavy tail, $\bar{F} \simeq x^{-\frac{1}{\gamma}}$	Pareto
$\gamma = 0$	Exponential tail, $\bar{F} \simeq e^{-x}$	Normal
$\gamma < 0$	Bounded, $\bar{F}(x) \underset{x \geq \tau}{=} 0$	Uniform

Table 2.1: *Relation between the survival function $\bar{F}$ and the tail index γ*

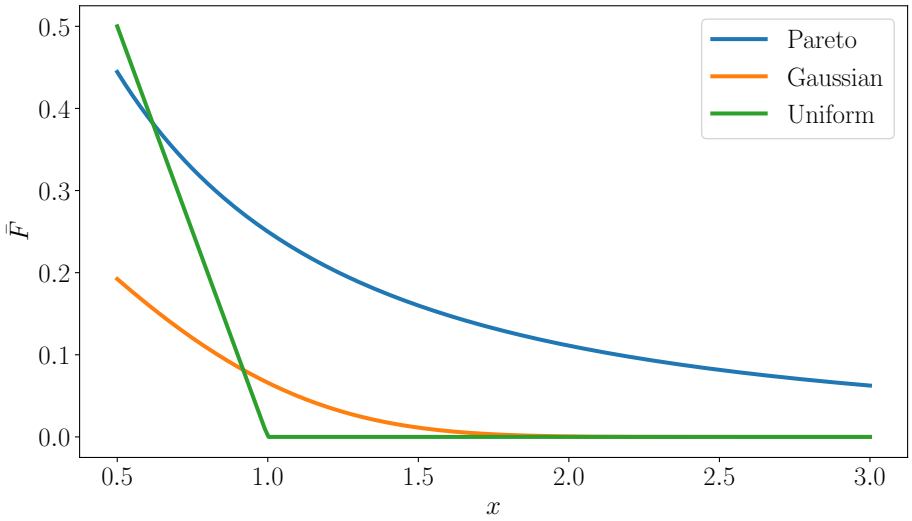


Figure 2.1: *Plot of survival functions with different tails*

Direct quantile estimation Let us recall that one have n independent and identically distributed observations $X_1, X_2 \dots X_n$. We can consider it as a sequence of records that can be cut in blocks of same size k: $\{X_1, \dots X_k\}, \{X_{k+1}, \dots X_{2k}\} \dots \{X_{n-k+1}, \dots X_n\}$. The maximum of each block can be computed, turning the raw records sequence into a *block maxima* sequence $M_{k,1}, M_{k,2} \dots M_{k,m}$, assuming $n = k \times m$. These maxima are m independent random variables with the same distribution as M_k. Considering that k is sufficiently large, a GEV distribution can be fit to them.

$$\mathbb{P}(M_k < x) = F^k(x) \simeq G_{\gamma,\mu,\sigma}(x)$$

Thus F can be retrieved to calculate quantiles. Now, if we look for z_q such that $\mathbb{P}(X > z_q) = q$, we have to solve:

$$q = 1 - \mathbb{P}(X < z_q) \simeq 1 - G^{1/k}_{\gamma,\mu,\sigma}(z_q).$$

By inverting the expression of $G_{\gamma,\mu,\sigma}$, it leads to

$$z_q \simeq \mu + \frac{\sigma}{\gamma}\left((-k \ln(1-q))^{-\gamma} - 1\right) \tag{2.2}$$

The drawback of this approach is that a great number of observations X_i are required. Furthermore, it assumes that the GEV parameters can be easily estimated, however an issue around the GEV fit is that the family has not some regularity properties. In particular, the support of the distribution depends on the parameters and such phenomenon leads especially to unusual properties for the maximum likelihood estimator.

2.2 The Peaks-Over-Threshold approach (POT)

As we said previously, looking at the block maxima to estimate the tail behavior implies some constraints. An alternative to directly consider the maximum is to look at the observations X_i that exceed an "high" threshold t. It allows to study the maximum through a Pareto distribution.

Distribution of the excesses The Pickands-Balkema-de Haan theorem [99, 6] is another view of the initial condition on F but its consequences allow to develop a method to estimate more efficiently the limit distribution of M_n.

THEOREM 2.2 (Pickands-Balkema-de Haan) *Let $\gamma \in \mathbb{R}$. $F \in \mathcal{D}_\gamma$ if and only if a function b exists such that for all x verifying $1 + \gamma x > 0$ we have:*

$$\frac{\bar{F}(t + b(t)\,x)}{\bar{F}(t)} \underset{t \to \tau}{\longrightarrow} (1 + \gamma x)^{-\frac{1}{\gamma}} \tag{2.3}$$

where τ represents the upper limit of the support of F (possibly infinite).

The limit introduced by the theorem 2.2 can be reformulated in terms of conditional probability. Let us introduce the survival function $\bar{F}_t(y) = \mathbb{P}(X - t > y \mid X > t)$ which denotes the probability of X to be higher than $y + t$ knowing that X is greater than t. We can notice that

$$\bar{F}_t(y\, b(t)) = \mathbb{P}\left(\frac{X-t}{b(t)} > y \;\middle|\; X > t\right) = \frac{\bar{F}(t + b(t)\, y)}{\bar{F}(t)}.$$

The Pickands-Balkema-de Haan theorem gives then the following tail estimation:

$$\bar{F}_t(y) = \mathbb{P}(X - t > y \mid X > t) \underset{t \to \tau}{\longrightarrow} \left(1 + \frac{\gamma y}{b(t)}\right)^{-\frac{1}{\gamma}}. \tag{2.4}$$

If $b(t)$ is interpreted as a scale parameter σ, it shows that F_t can be approximated to a generalized pareto distribution (GPD) with parameters γ, σ (the location μ being null).

POT quantile The previous result is not directly about F but below we will show how to retrieve a quantile with this approach. The behavior of F_t is known and allows to deduce those of F. First we have :

$$\bar{F}(t + y) = \bar{F}(t)\bar{F}_t(y)$$

So when t tends to τ, $\bar{F}_t(y)$ can be replaced by the limit given by (2.4)

$$\bar{F}(t + y) \simeq \bar{F}(t)\left(1 + \frac{\gamma y}{\sigma}\right)^{-\frac{1}{\gamma}}.$$

Let us write $x = t + y$, then $y = x - t$ and

$$\bar{F}(x) \simeq \bar{F}(t)\left(1 + \frac{\gamma(x-t)}{\sigma}\right)^{-\frac{1}{\gamma}}.$$

If we inverse the formula we get :

$$x \simeq t + \frac{\sigma}{\gamma}\left(\left(\frac{\bar{F}(x)}{\bar{F}(t)}\right)^{-\gamma} - 1\right)$$

Thus, if we are looking for the *extreme* quantile z_q such that $\bar{F}(z_p) = \mathbb{P}(X > z_q) = q$, we can replace x by $z_q = \bar{F}^{-1}(q)$ and get :

$$z_q \simeq t + \frac{\sigma}{\gamma}\left(\left(\frac{q}{\bar{F}(t)}\right)^{-\gamma} - 1\right) \tag{2.5}$$

POT procedure The previous quantile is really paramount in our problem however many things remain unestimated. Let us recall the context : we get an original sample $X_1, \dots, X_n$ and we can set a value for the threshold t. We note N_t the number of observations greater than t. First of all, as t is in the range of the X_i, $\bar{F}(t)$ can sensibly be estimated empirically :

$$\bar{F}(t) = \mathbb{P}(X > t) \simeq \frac{N_t}{n}$$

Finally the parameters γ and σ need to be estimated. In the case we get such estimators $\hat{\gamma}$ and $\hat{\sigma}$, the quantile can be computed through :

$$z_q \simeq t + \frac{\hat{\sigma}}{\hat{\gamma}} \left(\left(\frac{qn}{N_t} \right)^{-\hat{\gamma}} - 1 \right) \tag{2.6}$$

Finally, the POT estimation can be summed up as follows :

- Get a sample $X_1, \dots X_n$ and set a "high" threshold t
- Let N_t be the number of observations greater than t, compute $Y_1, Y_2 \dots Y_{N_t}$ the corresponding excesses ($Y = X - t$).
- Estimate $\hat{\gamma}, \hat{\sigma}$ through a GPD fit
- Compute z_q according to a given q (recall that q represent the probability for X to exceed z_q)

2.3 Parameters estimation

Estimating the GPD parameters is clearly the burning issue. We do know many techniques to compute such estimators but in our context their efficiency depend on the domain of the tail index γ. A review of their properties can be found in [125] and [80].

Method of moments The method of moments (MOM) try to fit the empirical moment to the theoretical ones. Indeed, the parameters often appear in the moment formula, thus by solving a system we can retrieve the best parameters $\hat{\gamma}$, $\hat{\sigma}$ to fit to a GPD. If $Y \sim \text{GPD}(\gamma, \sigma, \mu = 0)$, its moments are the following (the variance is defined only for $\gamma < \frac{1}{2}$):

$$\mathbb{E}(Y) = \frac{\sigma}{1-\gamma} \quad \text{and} \quad \text{Var}(Y) = \frac{\sigma^2}{(1-\gamma)^2(1-2\gamma)}.$$

With a sequence of observations $Y_1, Y_2 \dots Y_{N_t}$ we can easily compute the estimators of $\mathbb{E}(Y)$ and $\text{Var}(Y)$, noted E and V. Then by solving the system we get

$$\hat{\gamma}^{\text{MOM}} = \frac{1}{2}\left(1 - \frac{E^2}{V}\right) \quad \text{and} \quad \hat{\sigma}^{\text{MOM}} = \frac{1}{2}E\left(1 + \frac{E^2}{V}\right). \tag{2.7}$$

This method is but has a sense only for $\gamma < \frac{1}{2}$.

Probability-weighted moments The probability weighted moment (PWM) of random variable X, introduced by Greenwood *et al.* [43], are a generalization of the aforementioned moments. For a random variable Y, they are defined by :

$$M_{p,r,s} = \mathbb{E}\left[Y^p F(Y)^r \left(1 - F(Y)\right)^s\right]$$

As classical moments, the probability weighted moments of Y depend on the distribution parameters. In case of GPD, we can consider $p = 1$, $r = 0$ and $s = 0, 1, 2, \ldots$ leading to :

$$M_{1,0,s} = \frac{\sigma}{(s+1)(s+1-\gamma)}, \quad \gamma < 1$$

These moments are also empirically computable for a sequence $Y_1, Y_2 \ldots Y_{N_t}$. Choosing $s = 0$ and $s = 1$, it gives a system those solutions are :

$$\hat{\gamma}^{\text{PWM}} = 2 - \frac{\widehat{M}_{1,0,0}}{\widehat{M}_{1,0,0} - 2\widehat{M}_{1,0,2}}$$
$$\hat{\sigma}^{\text{PWM}} = \frac{2\,\widehat{M}_{1,0,0}\,\widehat{M}_{1,0,1}}{\widehat{M}_{1,0,0} - 2\widehat{M}_{1,0,2}}$$

The efficiency of both methods (MOM and PWM) has been studied by Hosking and Wallis [55]. They show that as well as not be defined on a large range these estimators are acceptable only on more restricted intervals. They have been improved but they still suffer in large sample.

Maximum likelihood estimate The maximum likelihood estimation remains a natural way to evaluate the parameters through observations. If $X_1, \ldots X_n$ are n independent realizations of the random variable X those density (noted f_θ) is parametrized by θ (possibly a vector), the likelihood function is defined by :

$$\mathcal{L}\left(X_1, \ldots X_n; \theta\right) = \prod_{i=1}^{n} f_\theta(X_i)$$

It represents joint density of these n observations. As $X_1, \ldots X_n$ are fixed in our context, we try to find the parameter θ such that the likelihood is maximized. It means that we are looking for the value of θ which makes our observations the "most probable". Let us apply to our context, the density function of a GPD is :

$$f_{\gamma,\sigma}(y) = \frac{1}{\sigma}\left(1 + \frac{\gamma}{\sigma}y\right)^{-\frac{1}{\gamma}-1}$$

With the observations $Y_1, Y_2 \ldots Y_{N_t}$, our likelihood function is :

$$\mathcal{L}\left(Y_1, \ldots Y_{N_t}; \gamma, \sigma\right) = \mathcal{L}\left(\gamma, \sigma\right) = \frac{1}{\sigma^n}\left[\prod_{i=1}^{N_t}\left(1 + \frac{\gamma}{\sigma}Y_i\right)\right]^{-\frac{1}{\gamma}-1}$$

In this case, maximizing $\log \mathcal{L}$ is more convenient. Practically, we try to minimize $-\log \mathcal{L}$ those expression is :

$$-\log \mathcal{L}(\gamma, \sigma) = N_t \log \sigma + \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_t} \log\left(1 + \frac{\gamma}{\sigma} Y_i\right) \tag{2.8}$$

Eventually, we have to notice the limit case $\gamma = 0$. By using Taylor series we can give the expression of the log-likelihood in that case.

$$\log\left(1 + \frac{\gamma}{\sigma} Y_i\right) \underset{\gamma \to 0}{=} \frac{\gamma}{\sigma} Y_i + o(\gamma)$$

so,

$$-\log \mathcal{L}(\gamma, \sigma) \underset{\gamma \to 0}{=} N_t \log \sigma + \left(1 + \frac{1}{\gamma}\right) \left(\frac{\gamma}{\sigma} \sum_{i=1}^{N_t} Y_i + o(\gamma)\right) = N_t \log \sigma + \frac{1}{\sigma} \sum_{i=1}^{N_t} Y_i + o(1)$$

Then when $\gamma = 0$, we can use the following expression

$$-\log \mathcal{L}(0, \sigma) = N_t \log \sigma + \frac{1}{\sigma} \sum_{i=1}^{N_t} Y_i. \tag{2.9}$$

2.4 Grimshaw's trick

Unfortunately, the minimization of the expression (2.8) must be done numerically. Precisely, this is an constraint minimization problem as we have to ensure that $\sigma > 0$ and that for all $i \in \{1, \dots N_t\}, 1 + (\gamma/\sigma) Y_i > 0$. These constraints can be put into the more canonical form $Ax > 0$ with

$$A = \begin{pmatrix} 1 & 0 \\ 1 & Y_1 \\ \vdots & \vdots \\ 1 & Y_{N_t} \end{pmatrix} \quad \text{and} \quad x = \begin{pmatrix} \sigma \\ \gamma \end{pmatrix}$$

These linear constraints can easily be taken into account by optimization algorithms, however another procedure given by Grimshaw [44] turns this two variables minimization problem into a univariate root search. Next, we will use $\ell(\gamma, \sigma) = -\log \mathcal{L}(\gamma, \sigma)$.

PROPOSITION 2.1 (Grimshaw) *Let (γ^*, σ^*) a solution of the system $\nabla \ell(\gamma, \sigma) = 0$. Then $x^* = \gamma^*/\sigma^*$ is a solution of the equation*

$$u(x^*)\, v(x^*) = 1, \tag{2.10}$$

with $u(x) = \dfrac{1}{N_t} \displaystyle\sum_{i=1}^{N_t} \frac{1}{1 + x Y_i}$ *and* $v(x) = 1 + \dfrac{1}{N_t} \displaystyle\sum_{i=1}^{N_t} \log(1 + x Y_i)$

Proof. First we can suppose that $\gamma \neq 0$ and we recall that $\sigma > 0$. We will use $x = \gamma/\sigma$. Let us detail the system $\nabla \ell(\gamma, \sigma) = 0$. In particular, we need the calculate the two partial derivatives.

$$\frac{\partial \ell}{\partial \gamma}(\gamma, \sigma) = \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_t} \frac{\frac{Y_i}{\sigma}}{1 + \frac{\gamma}{\sigma} Y_i} - \frac{1}{\gamma^2} \sum_{i=1}^{N_t} \log\left(1 + \frac{\gamma}{\sigma} Y_i\right)$$

So by multiplying by $\frac{\gamma^2}{N_t}$, the functions u and v appear.

$$\begin{aligned}
\frac{\gamma^2}{N_t} \times \frac{\partial \ell}{\partial \gamma}(\gamma, \sigma) &= (1+\gamma) \frac{1}{N_t} \sum_{i=1}^{N_t} \frac{\frac{\gamma}{\sigma} Y_i}{1 + \frac{\gamma}{\sigma} Y_i} - \frac{1}{N_t} \sum_{i=1}^{N_t} \log\left(1 + \frac{\gamma}{\sigma} Y_i\right) \\
&= (1+\gamma) \left(1 - \frac{1}{N_t} \sum_{i=1}^{N_t} \frac{1}{1 + \frac{\gamma}{\sigma} Y_i}\right) - \frac{1}{N_t} \sum_{i=1}^{N_t} \log\left(1 + \frac{\gamma}{\sigma} Y_i\right) \\
&= (1+\gamma)\,(1 - u(x)) - (v(x) - 1)
\end{aligned}$$

The second partial derivative is

$$\frac{\partial \ell}{\partial \sigma}(\gamma, \sigma) = \frac{N_t}{\sigma} + \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_t} \frac{-\frac{\gamma}{\sigma^2} Y_i}{1 + \frac{\gamma}{\sigma} Y_i}$$

Multiplying by $\frac{\gamma \sigma}{N_t}$, we get

$$\begin{aligned}
\frac{\gamma\,\sigma}{N_t} \times \frac{\partial \ell}{\partial \sigma}(\gamma, \sigma) &= \gamma - (1+\gamma) \left(1 - \frac{1}{N_t} \sum_{i=1}^{N_t} \frac{1}{1 + \frac{\gamma}{\sigma} Y_i}\right) \\
&= \gamma - (1+\gamma)\,(1 - u(x)) \\
&= (1+\gamma)\,u(x) - 1
\end{aligned}$$

The system $\nabla \ell(\gamma^*, \sigma^*) = 0$ with $\gamma^* \neq 0$ is equivalent to

$$\begin{cases} (1+\gamma)\,(1 - u(x^*)) - (v(x^*) - 1) &= 0 \\ (1+\gamma)\,u(x^*) - 1 &= 0 \end{cases}.$$

If we multiply the first equation by $u(x^*)$ it leads to

$$\begin{aligned}
& (1 - u(x^*)) - u(x^*)\,(v(x^*) - 1) &= 0 \\
\iff \qquad & u(x^*)\,v(x^*) &= 1.
\end{aligned}$$

Now, if we suppose that $\gamma^* = 0$, we have $x^* = 0$ and $u(x^*) = v(x^*) = 1$, so x^* is also a solution of the equation (2.10) when $\gamma^* = 0$. □

Naturally, the proposition 2.1 is an incentive to use the roots of $w : x \mapsto u(x)\,v(x) - 1$ as candidates to maximize the log-likelihood. Indeed by finding a root x^* of w, we can

retrieve $\gamma^* = v(x^*) - 1$ and $\sigma^* = \gamma^*/x^*$. Nevertheless, the solutions of this equation give only possible candidates for the minimum of ℓ, so in practice we need to get all the roots, to calculate the corresponding likelihood and keep the best tuple $(\hat{\gamma}, \hat{\sigma})$ as the final estimate.

We have to pay attention to how this numerical root search is done. Let us note

$$\overline{Y} = \frac{1}{N_t}\sum_{i=1}^{N_t} Y_i, \quad Y_{\max} = \max_{1\leq i\leq N_t} Y_i \quad \text{and} \quad Y_{\min} = \min_{1\leq i\leq N_t} Y_i.$$

The values $1 + xY_i$ must be strictly positives. As the Y_i are positive, we must find x^* on $\left(-\frac{1}{Y_{\max}}, +\infty\right)$. Grimshaw also calculates an upper-bound $x^*_{\max}$ for this root search given below (proposition 2.2).

PROPOSITION 2.2 *Let x^* be a solution of equation (2.10). Then*

$$x^* \leq x^*_{\max} = 2\frac{\overline{Y} - Y_{\min}}{(Y_{\min})^2}.$$

Proof. First, with the Jensen's inequality we have (the logarithm being concave)

$$v(x) = 1 + \frac{1}{N_t}\sum_{i=1}^{N_t} \log(1 + xY_i) \leq 1 + \log\left(\frac{1}{N_t}\sum_{i=1}^{N_t}(1 + xY_i)\right) = 1 + \log\left(1 + x\overline{Y}\right).$$

Then as $\forall i, Y_i \geq Y_{\min}$,

$$u(x) = \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{1}{1 + xY_i} \leq \frac{1}{N_t}\sum_{i=1}^{N_t}\frac{1}{1 + xY_{\min}} = \frac{1}{1 + xY_{\min}}$$

Now let us suppose that $x \geq x^*_{\max} \geq 0$. It is equivalent to

$$\begin{aligned} \frac{1}{2}xY_{\min}^2 &\geq \overline{Y} - Y_{\min} \\ \frac{1}{2}(xY_{\min})^2 &\geq x\overline{Y} - xY_{\min} \\ 1 + xY_{\min} + \frac{1}{2}(xY_{\min})^2 &\geq 1 + x\overline{Y} \end{aligned}$$

But $\exp(xY_{\min}) \geq 1 + xY_{\min} + \frac{1}{2}(xY_{\min})^2$ so $\log\left(1 + x\overline{Y}\right) < xY_{\min}$. We when $x \geq x^*_{\max}$ we have

$$\begin{aligned} u(x) &\leq \frac{1}{1 + xY_{\min}} \\ v(x) &\leq 1 + xY_{\min} \end{aligned}$$

therefore $w(x) = u(x)\,v(x) - 1 \leq 0$ in that case. A root of w cannot be found when $x \geq x^*_{\max}$. □

2.5 Practical GPD fit

In this section, we present our practical way to compute GPD parameters. Thanks to the proposition 2.2 the root search is restricted to a bounded interval

$$x^* \in (x_L, x_R) = \left(-\frac{1}{Y_{\max}}, 2\,\frac{\overline{Y} - Y_{\min}}{(Y_{\min})^2} \right).$$

In the proof of the proposition 2.1, we have seen that $x = 0$ is always a solution of the equation (2.10). So we need to find other possible roots in the two open intervals $(x_L, 0)$ and $(0, x_R)$. Several solutions can exist and Grimshaw has proposed a general method to sequentially find all of them. However we notice than in practice both intervals contains at most a single solution (a similar remark has also been made by Gimshaw in [44]).

Here, we present some additional properties of the function w. Under the assumption that each interval contains at most one root, these properties implies that there exist only two possibilities for the graph of w. First let us give its limits (proposition 2.3).

PROPOSITION 2.3 *The fonction w has the following limits*

$$\lim_{x \to x_L} w(x) = -\infty \quad \textit{and} \quad \lim_{x \to +\infty} w(x) = -1.$$

The proposition 2.4 gives now the behaviour of w close to 0.

PROPOSITION 2.4 *The following Taylor expansion holds*

$$w(x) \underset{x \to 0}{=} a(Y)\, x^2 + o(x^2) \tag{2.11}$$

where $a(Y) = \dfrac{1}{2}\left(\dfrac{1}{N_t} \displaystyle\sum_{i=1}^{N_t} Y_i^2 \right) - \left(\dfrac{1}{N_t} \displaystyle\sum_{i=1}^{N_t} Y_i \right)^2 = \dfrac{1}{2}\overline{Y^2} - \overline{Y}^2.$

Proof. We only need to compute the Taylor expansions of u and v.

$$\begin{aligned}
u(x) &\underset{x \to 0}{=} \frac{1}{N_t} \sum_{i=1}^{N_t} \left(1 - xY_i + (xY_i)^2 \right) + o(x^2) \\
&\underset{x \to 0}{=} 1 - x\overline{Y} + x^2 \overline{Y^2} + o(x^2) \\
v(x) &\underset{x \to 0}{=} 1 + \frac{1}{N_t} \sum_{i=1}^{N_t} \left(xY_i - \frac{1}{2}(xY_i)^2 \right) + o(x^2) \\
&\underset{x \to 0}{=} 1 + x\overline{Y} - \frac{1}{2} x^2 \overline{Y^2} + o(x^2).
\end{aligned}$$

So by multiplying the two series, we get

$$\begin{aligned}
w(x) - 1 = u(x)\, v(x) &\underset{x \to 0}{=} \left(1 - x\overline{Y} + x^2 \overline{Y^2} + o(x^2) \right) \left(1 + x\overline{Y} - \frac{1}{2} x^2 \overline{Y^2} + o(x^2) \right) \\
&\underset{x \to 0}{=} 1 + x\overline{Y} - \frac{1}{2} x^2 \overline{Y^2} - x\overline{Y} - x^2 \overline{Y}^2 + x^2 \overline{Y^2} + o(x^2) \\
&\underset{x \to 0}{=} 1 + x^2 \left(\frac{1}{2}\overline{Y^2} - \overline{Y}^2 \right) + o(x^2).
\end{aligned}$$

□

REMARK 2.2 *$a(Y)$ involves two terms which actually are the common estimators of the expected value and the variance. To ease the notations we may use:*

$$\mathbb{E}(Y) = \overline{Y} = \frac{1}{N_t}\sum_{i=1}^{N_t} Y_i \quad \textit{and} \quad \mathrm{Var}(Y) = \overline{Y^2} - \overline{Y}^2.$$

With these notations, $a(Y) = \frac{1}{2}\left(\mathrm{Var}(Y) - \mathbb{E}(Y)^2\right)$

We can notice that w can be approached by a parabola around $x = 0$. The key point is that the orientation of the parabola is given by the sign of $a(Y)$. Taking into account the propositions 2.3 and 2.4, the function w has two possible shapes under the assumption that each interval contains at most one root. Instead of giving formal proofs, we rather illustrate the two possible behaviors of w on figure 2.2.

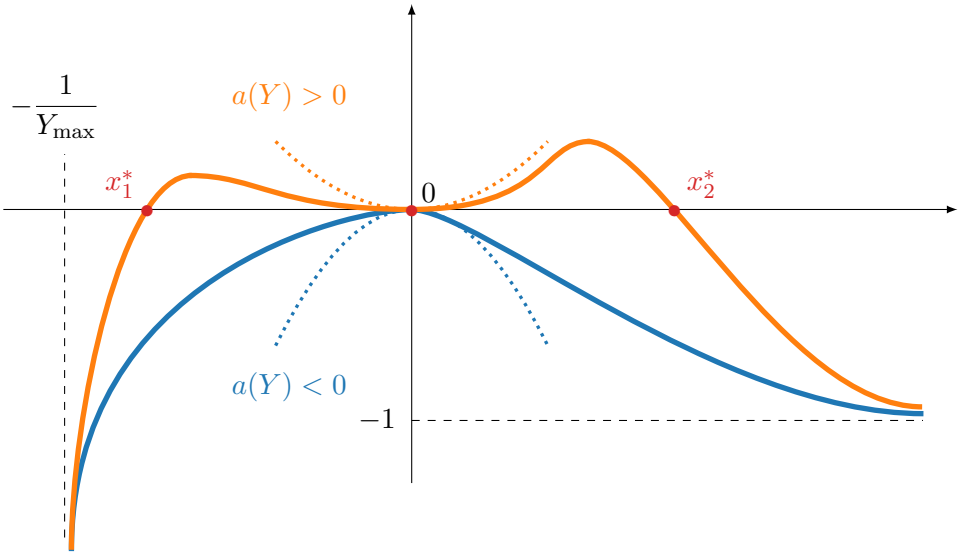


Figure 2.2: *Two possible shapes of w under the assumption that the function has a limited number of roots.*

The sign of $a(Y)$ gives the local shape of w around 0. When $a(Y) < 0$ (blue curve), the function w is locally concave and then no root lie in any intervals. Indeed, if we assume that a root exist in an interval, another one exists in the same interval too. When $a(Y) \geq 0$ (orange curve), the function w is locally convex then two roots x_1^*, x_2^* necessarily exist in each interval to respect the limit of w.

This observation leads to a approximate procedure to find the GPD parameters from the excesses Y_i (see algorithm 1). First, we need to compute the minimum of ℓ when $\gamma = 0$ (line 3). In this case, the expression (2.9) must be used and we can easily show that its minimum is reached when $\sigma = \mathbb{E}(Y) = \overline{Y}$. Then we check the orientation of

the parabola so as to decide if other roots than 0 exist (line 5). If there is no root we keep $(0, \overline{Y})$ as the best configuration (line 22). When roots must be found, a classical root search algorithm can be used, like the Bisection method, the Newton's method or the Brent's method [11] for instances. Obviously, we need to avoid both the origin (root already known) and x_L (where w is not defined), so we introduce a small parameter ϵ. We can also wonder whether the classical condition $w(a)\,w(b) < 0$ is respected (where a and b are the interval bounds) to perform the root search. On both intervals, we can merely decrease ϵ if the condition is not respected. If several roots have been found, their corresponding likelihoods are compared. The best configuration is naturally returned.

Algorithm 1 GPD fit

1: **procedure** GRIMSHAW($Y, \epsilon > 0$)
2: $V \leftarrow \mathrm{Var}(Y), E \leftarrow \mathbb{E}(Y)$
3: $\ell_{\min} \leftarrow \ell\left(0, \overline{Y}\right)$ ▷ $x = 0$ is always solution
4: $a \leftarrow V - E^2$ ▷ Check the orientation of the parabola
5: **if** $a \geq 0$ **then**
6: $x_1^* \leftarrow$ ROOTSEARCH($w, x_L + \epsilon, -\epsilon$)
7: $\gamma_1 \leftarrow v(x^*) - 1; \sigma_1 \leftarrow \gamma_1 / x^*$
8: $\ell_1 \leftarrow \ell(\gamma_1, \sigma_1)$
9: $x_2^* \leftarrow$ ROOTSEARCH(w, ϵ, x_R)
10: $\gamma_2 \leftarrow v(x^*) - 1; \sigma_2 \leftarrow \gamma_2 / x^*$
11: $\ell_2 \leftarrow \ell(\gamma_2, \sigma_2)$
12: **if** $\ell_1 < \ell_2$ **then**
13: $\gamma^*, \sigma^* \leftarrow \gamma_1, \sigma_1$
14: $\ell^* \leftarrow \ell_1$
15: **else**
16: $\gamma^*, \sigma^* \leftarrow \gamma_2, \sigma_2$
17: $\ell^* \leftarrow \ell_2$
18: **end if**
19: **if** $\ell^* < \ell_{\min}$ **then**
20: **return** γ^*, σ^*
21: **else**
22: **return** $0, E$
23: **end if**
24: **end if**
25: **return** $0, E$
26: **end procedure**

2.6 Conclusion

Extreme Value Theory is a powerful framework to analyze the distribution of extreme events. Its core theorems allow to infer the tail of any distribution without strong

assumption. Thus it is possible to compute probabilities and quantiles in the extreme regions where we may lack observations. However, estimating the best model for the tail of a distribution is very hard. Thanks to the previous work of Grimshaw, the maximum likelihood estimation is eased and it leads us to build a simple procedure to perform the fit. The GPD fit will be a key element of the anomaly detection algorithm presented in the next chapter.

3

The SPOT algorithm

The extreme value theory (EVT), through the Peaks-Over-Thresholds (POT) approach, gives us a way to estimate z_q such that $\mathbb{P}(X > z_q) < q$ without any strong assumption on the distribution of X and without any clear knowledge about its distribution.

In this chapter we adapts this method to work with streams, building then a new streaming univariate anomaly detector called SPOT (Streaming Peaks-Over-Threshold). SPOT embeds the strengths of the parametric statistical methods without the weaknesses. Thanks to EVT, the statistical model used is indeed very general making SPOT not suffer from the distribution assumption. Moreover, we propose a more general variant, called DSPOT, which takes into account the drift within data streams.

Through detailed experiments on synthetic and real data, we show that our approach is accurate for detecting outliers, is computationally efficient, and for DSPOT reacts quickly to any change in the stream. For instance, we decide to test our algorithm on incoming streams without knowledge on their distribution. We show that we detect very efficiently and accurately: (i) network SYN attacks on a labeled data stream and (ii) peaks that allows to take decision on stock market (quickly react for buying or selling shares). Our experiments also confirm the EVT theory with accuracy and fast convergence.

First we present the initialization step which computes a threshold z_q from n initial observations $X_1, \dots X_n$. Then, we detail our two streaming algorithms which update z_q with the incoming data and use it as a decision bound. Eventually we experiment them on synthetic and real world datasets.

In the next paragraphs, we use some results presented in the previous chapter therefore some points are not more detailed. If necessary, the reader should then refer to chapter 2.

3.1 Initialization

Let us sum up the basic idea of the SPOT algorithm. We want to compute and update along time a threshold z_q verifying $\mathbb{P}(X > z_q) < q$ where q is user-defined. The parameter q is paramount and corresponds to the probability of an anomaly.

Initially, we have n observations $X_1, \dots X_n$, and we have set q. The goal is to compute a first threshold z_q. The figure 3.1 shows what we do on this initial batch (calibration). First we compute a high threshold t (e.g. a high empirical quantile practically), then we retrieve the *peaks* (the excesses over t) and we fit a GPD (Generalized Pareto Distribution) to them. So that we infer the distribution of the extreme values and we can compute an initial threshold z_q. The latter value will be used to flag anomalies afterwards.

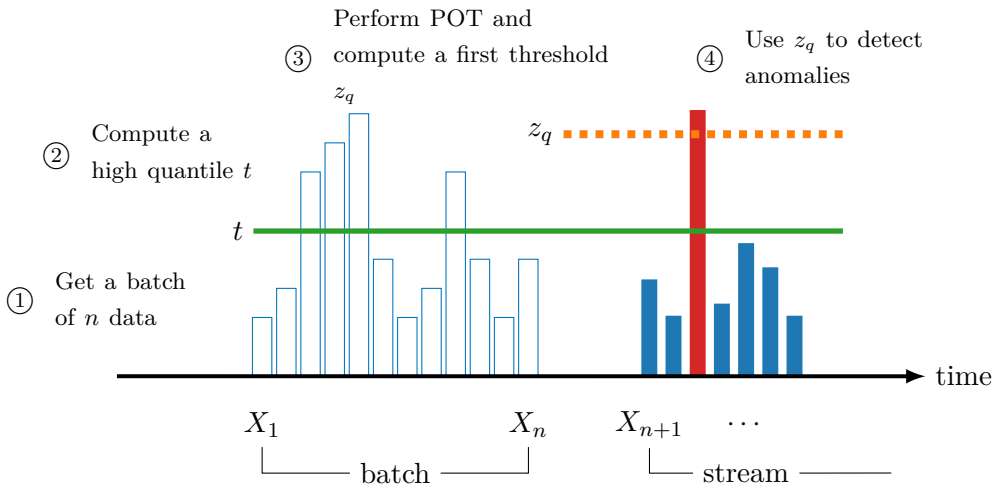


Figure 3.1: *Anomaly detection overview (calibration)*

This initialization step is summarized in the algorithm 2. The choice of t (line 2), called the *peak threshold*, will be detailed in 3.3. As we said, we take the empirical quantile of the batch $\{X_1, \dots X_n\}$ at level p_t, given as a parameter (p_t is then called *level*). The set $\mathbf{Y}_t$ is the *peaks set* where we store the observed excesses over t. The GPD fit is performed with the algorithm 1 presented in 2.4 and then we can compute z_q (line 5) with equation (2.6).

Algorithm 2 POT (Peaks-over-Threshold)

1: **procedure** POT($X_1, \dots X_n, q, p_t$)
2: $t \leftarrow \widehat{F}^{-1}(p_t)$
3: $\mathbf{Y}_t \leftarrow \{X_i - t \,|\, X_i > t\}$
4: $\hat{\gamma}, \hat{\sigma} \leftarrow$ Grimshaw($\mathbf{Y}_t$)
5: $z_q \leftarrow$ CalcThreshold($q, \hat{\gamma}, \hat{\sigma}, n, N_t, t$)
6: **return** z_q, t
7: **end procedure**

3.2 Finding anomalies in a stream

The POT primitive returns a threshold z_q which we use to define a "normality bound" (figure 3.1). In our streaming algorithms the POT primitive (algorithm 2) is used as an initialization step. The POT primitive may be seen as a *training* step but this is partly wrong because the initial batch $X_1, \dots X_n$ is not labeled and is not considered as a ground truth in our algorithm. The initialization is more a *calibration* step. Our streaming anomaly detector uses the next observations to both detect anomalies and refine the anomaly threshold z_q.

Stationary case The way how the POT estimate is built is really stream-ready. As we do not have to store the whole time series (only the peaks), it requires low memory so we can use it in a stream. However, the stream must contain values from the same distribution, so this distribution cannot been time-dependent (what we call *stationary*). In case of time-dependency, we will show that our algorithm can be adapted to drifting cases (see 3.2).

The algorithm 3 details the SPOT algorithm. The aim is to detect abnormal events in a stream $(X_i)_{i>0}$ in a blind way (without any knowledge about the distribution). Firstly, we perform a POT estimate (line 3) on the n first values ($n \sim 1000$) and we get an initial threshold z_q (this is the initialization we explained in the previous paragraph). Then for all the next observed values we can flag the events or update the threshold. If a value exceeds our threshold z_q (line 6) then we consider it as abnormal (we can retrieve this anomaly in a list $\mathbf{A}$). The anomalies are not taken into account for the model update. In the other cases, either X_i is greater than the initial threshold (peak case, line 8) either it is a "common" value (normal case). In the peak case, we add the excess to the peaks set and we update the threshold z_q (lines 9 to 14).

In this algorithm we perform the maximum number of threshold updates but it is possible to do it off-line at fixed time interval. Of course we illustrate the principle only with upper-bound thresholds but the method is the same for lower-bound ones and we can even combine both.

Drifting case SPOT assumes that the distribution of the X_i does not change over time but it might be restrictive. For instance, a mid-term seasonality cannot be taken into

Algorithm 3 SPOT (Streaming POT)

1: **procedure** SPOT($(X_i)_{i>0}, q, n, p_t$)
2: $\mathbf{A} \leftarrow \emptyset$ ▷ set of the anomalies
3: $z_q, t \leftarrow \text{POT}(X_1, \dots X_n, q, p_t)$ ▷ calibration
4: $k \leftarrow n$
5: **for** $i > n$ **do**
6: **if** $X_i > z_q$ **then** ▷ anomaly case
7: Add (i, X_i) in $\mathbf{A}$
8: **else if** $X_i > t$ **then** ▷ real peak case
9: $Y_i \leftarrow X_i - t$
10: Add Y_i in $\mathbf{Y}_t$
11: $N_t \leftarrow N_t + 1$
12: $k \leftarrow k + 1$
13: $\hat{\gamma}, \hat{\sigma} \leftarrow \text{GRIMSHAW}(\mathbf{Y}_t)$ ▷ update the model
14: $z_q \leftarrow \text{CALCTHRESHOLD}(q, \hat{\gamma}, \hat{\sigma}, k, N_t, t)$ ▷ update the threshold
15: **else** ▷ normal case
16: $k \leftarrow k + 1$
17: **end if**
18: **end for**
19: **end procedure**

account, making local peaks undetectable. In this section we overcome this issue by modeling an average local behavior and applying SPOT on relative gaps.

We propose Drift SPOT (DSPOT) which makes SPOT run not on the absolute values X_i but on the relative ones. We use the variable change $X'_i = X_i - M_i$ where M_i models the local behavior at time i (see figure 3.2). In our implementation we use a moving average $M_i = (1/d) \cdot \sum_{k=1}^{d} X^*_{i-k}$ with $X^*_{i-1}, \dots X^*_{i-d}$ the last d "normal" observations (so d is a window size parameter). In this new context we assume that the local variations X'_i come from a same stationary distribution (the hypothesis assumed for X_i in SPOT is now assumed for X'_i).

This variant uses an additional parameter d, which can be viewed as a window size. The distinctive features of this window (noted W^*) are the following: it might be non continuous and it does not contain abnormal values.

The algorithm 4 shows our method to capture the local model and perform SPOT thresholding on local variations. It contains some additional steps so as to compute variable changes. For these stages, we principally use a sliding windows over normal observations W^* (lines 3, 7, 24 and 28) in order to calculate a local normal behavior M_i (lines 4, 8, 25 and 29) through averaging. We logically update the local behavior only in normal or peak cases (lines 25 and 29).

We can retrieve sequentially the "real" extreme quantiles by adding M_i to the calculated z_q. Such a choice to model the local behavior is a very efficient way to adapt SPOT to drifting contexts. As mentioned in the previous paragraph, DSPOT can be

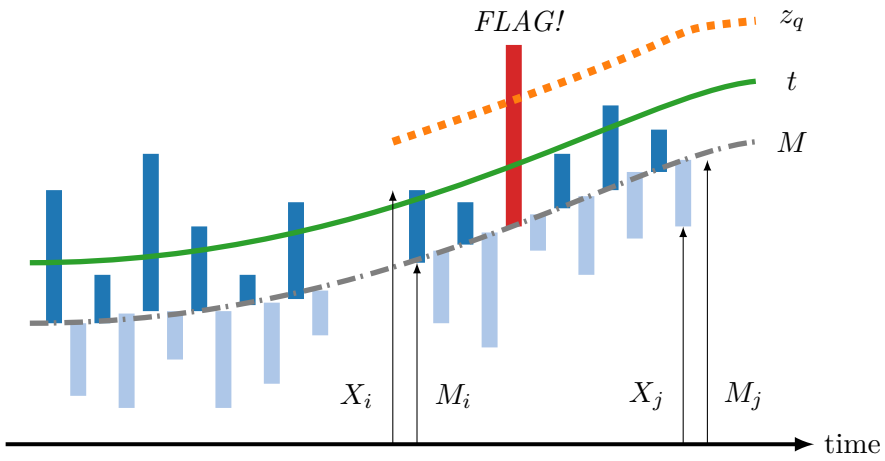


Figure 3.2: *Anomaly detection with drift*

adapted to compute upper and lower bounds.

3.3 Practical aspects

In this section we give details about the parameters used by the algorithms. SPOT has a single main parameter q which sets the decision threshold z_q. However, the two side parameters p_t (the level) and n (the size of the calibration batch) must also discussed for a practical purpose. Both have not a huge impact on the algorithm once they respect some conditions.

The variable t represents the bias-variance trade-off. In practice, the value of t must be "high" enough to only keep the peaks which are in the tail of the distribution. The higher is t, the more relevant will be the GPD fit (low bias). However, if t is too high, the peaks set $\mathbf{Y}_t$ would be little filled in, so the model would be more variable (high variance). Naturally, the important condition is that t must be lower than z_q, meaning that $F(t) = \mathbb{P}(X \leq t)$, the probability associated to t, must be lower than $1-q$. That is why, the value of $p_t = \widehat{F}(t)$ is tuned rather than the absolute value of t, and we generally set $p_t \geq 98\%$. We have to note that p_t has a sense only during the calibration step since t is computed from the empirical distribution.

There is a close link between p_t and n, the size of the initial batch to perform the calibration. During the calibration, the initial fit will use the $1 - F(t)$ highest peaks, so $k = (1 - p_t) \times n$ values. We have mentioned that t must be high (i.e. p_t close to 1) to reduce the bias but we have also to ensure that k will be enough to avoid a too large variance (namely a fit with too few values). In practice, several dozens values as needed to perform a reliable fit, therefore we recommend to set n and $F(t)$ such that $(1 - p_t)\ n \geq 10$.

The two aforementioned conditions lead to a "minimal" but reliable configuration: $n = 1000$ and $p_t = 0.98$. If more data are available for the calibration, we could advise

Algorithm 4 DSPOT (Streaming POT with drift)

procedure DSPOT($(X_i)_{i>0}, d, q, n, F(t)$)
  $\mathbf{A} \leftarrow \emptyset$ ▷ set of the anomalies
  $W^* \leftarrow [X_1, \dots X_d]$ ▷ Last d normal values
  $M_{d+1} = \overline{W^*}$ ▷ Local model with depth d
  for $i \in [\![d+1, d+n]\!]$ **do**
    $X'_i = X_i - M_i$
    $W^* \leftarrow [X_{i-d+1}, \dots X_i]$
    $M_{i+1} \leftarrow \overline{W^*}$
  end for
  $z_q, t \leftarrow \mathrm{POT}(X'_{d+1}, \dots X'_{d+n}, q, F(t))$
  $k \leftarrow n$
  for $i > d+n$ **do**
    $X'_i = X_i - M_i$ ▷ variable change
    if $X'_i > z_q$ **then** ▷ anomaly case
      Add (i, X_i) in $\mathbf{A}$
      $M_{i+1} \leftarrow M_i$ ▷ no update
    else if $X'_i > t$ **then** ▷ real peak case
      $Y_i \leftarrow X'_i - t$
      Add Y_i in $\mathbf{Y}_t$
      $N_t \leftarrow N_t + 1$
      $k \leftarrow k + 1$
      $\hat{\gamma}, \hat{\sigma} \leftarrow \textsc{Grimshaw}(\mathbf{Y}_t)$
      $z_q \leftarrow \textsc{CalcThreshold}(q, \hat{\gamma}, \hat{\sigma}, k, N_t, t)$
      $W^* \leftarrow W^*[1:] \cup X_i$ ▷ window slide
      $M_{i+1} \leftarrow \overline{W^*}$ ▷ update of the local model
    else ▷ normal case
      $k \leftarrow k + 1$
      $W^* \leftarrow W^*[1:] \cup X_i$ ▷ window slide
      $M_{i+1} \leftarrow \overline{W^*}$ ▷ update of the local model
    end if
  end for
end procedure

to increase both n and p_t, keeping the constraint on the product $(1 - p_t)\, n$.

3.4 Experiments on stationary streams

In this section we apply both our algorithms SPOT and DSPOT on several contexts. First, we compare the computed threshold z_q with the theoretical ones through experiments on synthetic data. Then we use real world datasets from several fields (network, physics, finance) to highlight the properties of our algorithms.

SPOT reliability In this section, we compare our computed threshold z_q to the theoretical one. In other words, we want to check if z_q is the desired threshold (which verifies $\mathbb{P}(X > z_q) < q$). In the same time we evaluate the impact of the number of observations n in the initial batch.

In the following experiments we set $q = 10^{-3}$ and we run SPOT on Gaussian white noises of 15000 values, i.e. 15000 independent values from a standard normal distribution ($\mu = 0, \sigma^2 = 1$). For different values of n (300, 500, 1000, 2000, 5000 and 10000), we run SPOT 100 times and we retrieve the averaged error made in comparison to the theoretical threshold:

$$\text{error rate} = \left| \frac{z^{\text{SPOT}} - z^{\text{th}}}{z^{\text{th}}} \right|.$$

Here, z^{th} is the quantile of the standard normal distribution at level $1-q$, so $z^{\text{th}} \simeq 3.09$. The figure 3.3 presents the results.

First of all the curves show that, for all initial batch sizes n, the error is low and decreases when the number of observations increases. It means that the computed threshold z^{SPOT} is close to the theoretical one and tends to it.

Secondly, we have to notice that the error curves all converge to the same value regardless of n. Therefore, n is not a paramount parameter. In our experiments, we just have to ensure that n is not too small, otherwise the initialization step is likely to fail because of a lack of peaks to perform the GPD fit. Generally, we use $n \simeq 1000$.

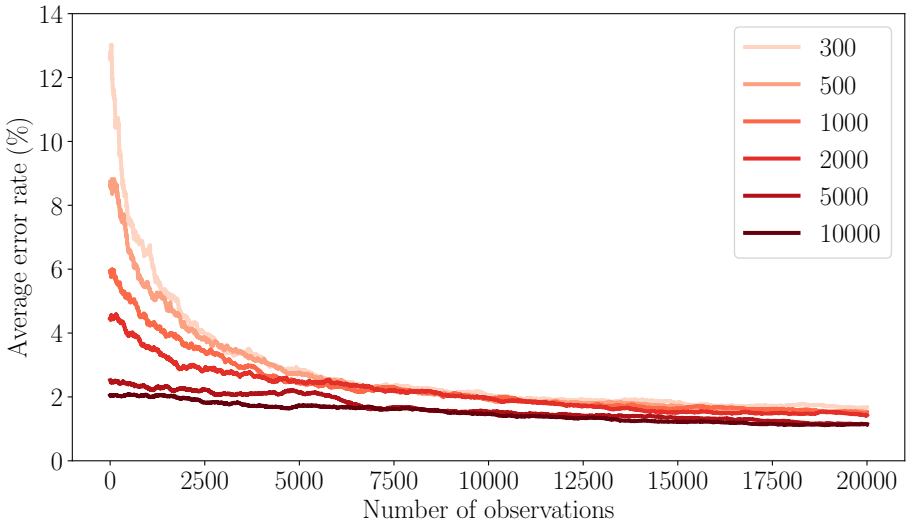


Figure 3.3: *Error rate with the number of observations according to the batch size n*

Intrusion detection example SPOT computes a robust threshold estimation (z_q): the more data we monitor, the more accurate the estimation is. So having a lot of data from

the same and unknown distribution, SPOT can gradually adapt this threshold in order to detect anomalies. Cyber-security is a typical field where such configurations appear.

To test our algorithm we use real data from the MAWI repository which contains daily network captures (15 minutes a day stored in a `.pcap` file). In these captures, MAWIlab [36] finds anomalies and labels them with the taxonomy proposed by Mazel *et al.* [86]. The anomalies are referred through detailed patterns. To be close to real monitoring systems we converted raw `.pcap` files into NetFlow format, which aggregates packets and retrieves meta-data only, and is commonly used to measure network activity. Then we labeled the flows according to the patterns given by the MAWIlab. In this experiment we use the two captures from the 17/08/2012 and the 18/08/2012.

Classical attacks are network scans where many SYN packets are sent in order to find open and potentially vulnerable ports on several machines. A relevant feature to detect such attack is the ratio of SYN packets in a given time window [46]. From our NetFlow records we compute this feature on successive 50 ms time windows and we try to find extreme events. To initialize SPOT we use the last 1000 values of the 17/08 record and we let the algorithm working on the 18/08 capture.

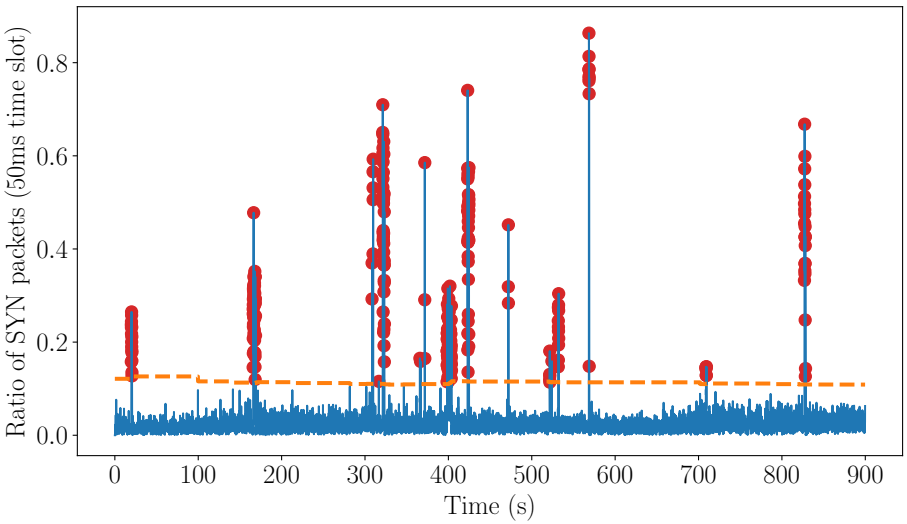


Figure 3.4: *SYN flood detection with* $q = 5.10^{-4}$

The figure 3.4 shows the alerts triggered by SPOT (red circles). We recall that each point represents a 50 ms window gathering several flows (possibly benign and malicious). The computed threshold (dashed line) seems nearly constant but this behavior is due to the stability of the measure we monitor (SPOT has quickly inferred the behavior of the feature). By flagging all the flows in the triggered windows, we get a true positive rate equal to 86% with less than 4% of false positives.

The parameter q as a false-positive regulator In the previous section we noticed that the size of the initial batch n is not an important parameter insofar as it does not affect the overall behavior of SPOT. Here, we study the impact of the main parameter q on the MAWI dataset.

On the figure 3.5, the ROC curve shows the effect of q on the False Positive rate (FPr). Values of q between 10^{-3} and 10^{-4} allow to have a high TPr while keeping a low FPr: this leaves some room for error when setting q.

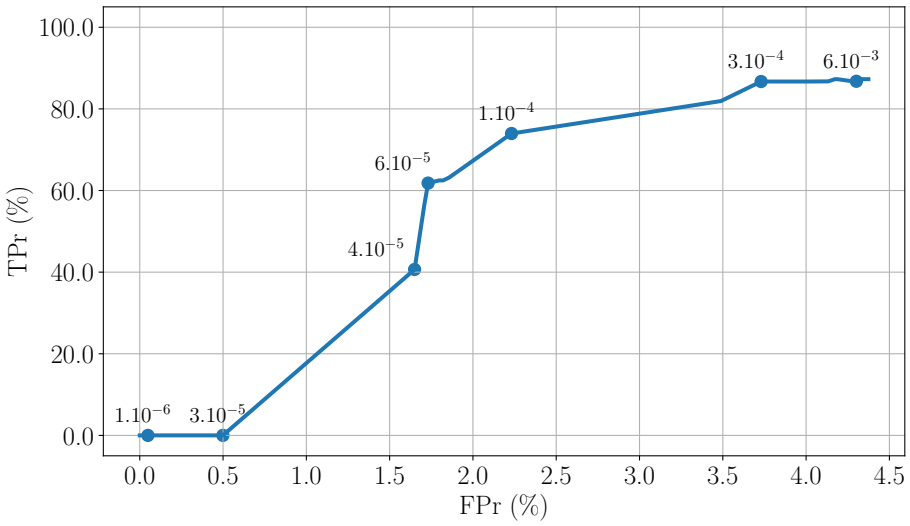


Figure 3.5: *ROC curves on MAWI dataset (markers give the corresponding value of q)*

3.5 Experiments on drifting streams

Measure of the magnetic field To show the wide variety of fields on which we can use DSPOT, we apply our algorithm on astrophysics measures from the SPIDR (Space Physics Interactive Data Resource[1]). SPIDR is an online platform which stores and manages historical space physics data for integration with environment models and space weather forecasts.

Particularly we use a dataset available on Comp-Engine Time Series[2] which gathers some physical measures taken by the ACE satellite between 1/1/1995 and 1/6/1995. In the figure 3.6 we monitor a component of the magnetic field (in nT) during several minutes.

This time series is very noisy and contains different complex behaviors. We calibrate DSPOT with the $n = 2000$ first values and we run on the 15801 others with $q = 5.10^{-4}$ and $d = 400$ ($p_t = 0.99$). The results are depicted on the figure 3.6.

[1] `http://spidr.ionosonde.net/spidr/home.do`

[2] `http://www.comp-engine.org/timeseries/time-series_data/data-17481/`

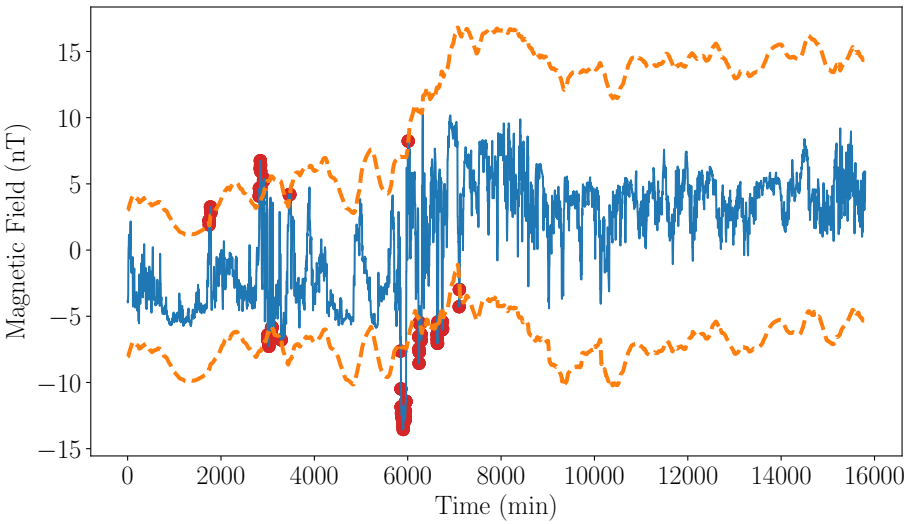


Figure 3.6: *DSPOT run with* $q = 5.10^{-4}, d = 400$

At the first glance, the bounds are following the signal and some alarms are triggered by high peaks. After 9000 minutes, the upper bound seems higher than expected.

To understand why, let us divide the signal into two parts: before and after 8000 minutes. Before 8000, we can observe that "peaks" lean upwards the trend although the opposite phenomenon appears after 8000. During these 8000 first minutes, DSPOT learns that peaks may lean upwards the trend and it keeps this information in memory. Hence after 8000 minutes, the upper bound stays high in order to accommodate for possible peaks above the trend. DSPOT has this behavior because it keeps a global memory of all peaks encountered. If it was not desired, un easy modification is to keep only the last k peaks, for a fixed k. In particular, our libraries implement such a "bounded mode" so as to set the maximum number of peaks to stored. It then add a memory depth which

Stock prices On Thursday the 9th of February 2017, an explosion happened at Flamanville nuclear plant, in northern France. This power plant is managed by EDF, a French electricity provider. The incident was not in the nuclear zone and did not hurt people[3]. This incident was officially declared at 11:00 a.m. making the EDF stock price fall down. This recent event encouraged us to test DSPOT on EDF stock prices.

Obviously, retrieving financial data with high resolution is not within ours grasp. However, some websites like Google Finance[4] propose intraday financial data with a record per minute. Google Finance keeps these records during 15 days, so we retrieve

[3] `http://www.webcitation.org/6oAxqoFkf`
[4] `https://www.google.com/finance`

the records from the 6th to the 8th of February 2017 for calibration (1062 values) and ran DSPOT on the explosion day (379 values).

On figure 3.7, we notice that DSPOT follows the average behavior and flags the drop around 11:00 a.m. This may help a trading system to quickly take actions or warn experts.

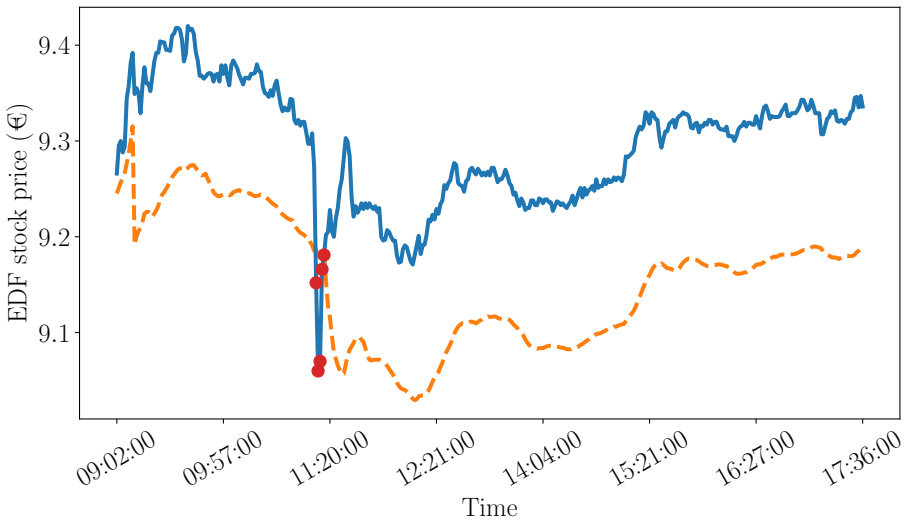


Figure 3.7: *DSPOT run with* $q = 10^{-5}, d = 10$ *on EDF stock prices (2017-02-09)*

3.6 Implementation

Here we give some details about the implementations of (D)SPOT, which are mainly available at `https://asiffer.github.io/libspot/`.

Available implementation The original library (called `libspot`) is in `C++`. It is then possible to use it directly but `python3` bindings have also been made to use it at a higher level. These two implementations are distributed through `debian` packages:

- `libspot`, `libspot-dev` (headers)
- `python3-libspot`

Thus, they are downloadable with the `apt` package manager. The `python3` code is also available on PyPI, so through the `pip` package manager. Moreover we have recently provides a `Go` implementation available at `https://github.com/asiffer/gospot`.

Performances To test the performances of the different SPOT implementations, we run each of them on Gaussian white noises of different length (number of observations N). For each N, the algorithm is run 200 times and the average running time is retrieved. For this purpose, we use a common configuration: $q = 10^{-4}, n = 1000$ and $p_t = 0.99$. All these tests have been made on a laptop with an Intel i5-8250U CPU @ 1.60GHz (8 cores) and 16 GB RAM. The results are presented on figure 3.8.

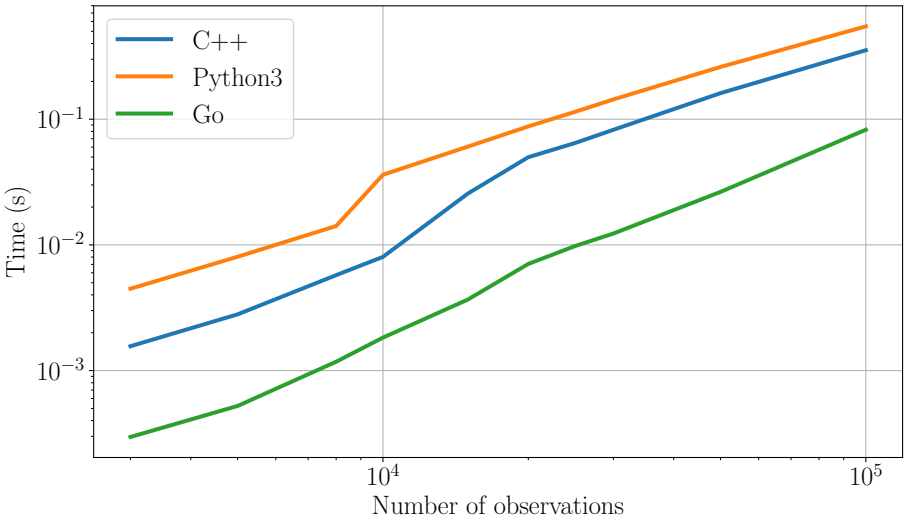


Figure 3.8: *Spot implementations performances*

We can notice that they are all efficient since they are able to tackle several tens of thousands observations in less than a second. We also observe the linear complexity of the algorithm which is a requested feature to treat streaming data. We may wonder why the `Go` implementation if faster than the `C++` version: actually, the `C++` library is not compiled with the optimizations while the `Go` compiler does it by default.

3.7 Conclusion

This chapter has presented a novel approach to detect outliers in high throughput numerical time series. The key points of our approach is that it does not assume the data distribution, and it does not require manually set thresholds. It adapts on multiple and complex contexts, learning how the interest measure behaves. It achieves these results by using the Extreme Values Theory (EVT). To the best of our knowledge, it is the first time EVT has been used to detect outliers in streaming data.

There are two kinds of perspectives. From the theoretical side, in this work we only exploited a small part of EVT, we would like to extend this work to the multivariate case. Indeed, (D)SPOT is only able to address univariate time series which limits its direct application. The main hindrance is that EVT does not well generalize to higher

dimensions, mainly for the following reason: the definition of "extreme value" is not as explicit as in the univariate case.

From the practical side, one of the immediate application of our approach is *automatic thresholding*: it can provide thresholds with strong statistical guarantees that can adapt to the evolution of a data stream. We would like to explore the use of our approach as a building block into more complex systems that require thresholding.

Finally, a solution to design a more general anomaly detector is to combine SPOT with a scoring algorithm. The role of the latter would be to accurately weight multivariate observations (according to their normality/abnormality) while SPOT task would be to automatically threshold these scores to discriminates the outliers. In the next chapter, we will present an intrusion detection systems (IDS) we have been developed which monitors network features with SPOT.

4

Application to intrusion detection

Both research and industrial actors are aware that our progresses in AI increase our overall knowledges and abilities. They provide much understanding about processes and behaviours through data mining and tasks formerly hard then become trivial thanks to machine learning. However, all these cutting-edge methods seem to elude the cyber-security field. Paradoxically, data are plentiful and our expertise is very advanced. Actually, the aims are more ambitious and the challenges are more constrained.

Two intrusion detection approaches exist: rule-based and anomaly-based. As they are efficient and mature, rule-based solutions (like Snort[1], Zeek[2] or Suricata[3]) are the most widely deployed but anomaly detection tries to extend beyond the scope of attack signature matching. Indeed, the latter has several weaknesses: building signatures is lengthy and laborious, the payload is likely to be inspected although it is an expensive task which becomes deprecated with the wide use of end-to-end encryption and

[1] https://www.snort.org/
[2] Previously known as Bro, https://www.zeek.org/
[3] https://suricata-ids.org/

eventually, rule-based methods are static thus new attacks cannot be detected. Anomaly-based techniques aim to build a model of *normality* (in a supervised or unsupervised way) in order to discriminate abnormal observations (often considered as attacks).

Much work has been done to develop anomaly detectors and with the help of cutting edge algorithms, smarter IDS have been proposed [90]. Despite the feeling that research is far from the industrial context, several works attempt to solve real-world issues.

One of these problems is what is generally called *parametrization*. Anomaly detectors can be very powerful provided that they are well configured. So they generally need a great amount of expertise to be set up in a specific network, in particular to set decision thresholds. Actually, they often have no meaning and require a time-consuming fine-tuning step to avoid too many false positive.

First, we will present why such a distrust of Machine Learning/Data Mining (ML/DM) techniques for intrusion detection. Then, two relevant anomaly-based IDS will be detailed, showing that several works try to tackle the cyber-security challenges. Finally, we will present `netspot`, an IDS we developed powered by the SPOT algorithm. Despite its simplicity, we will see that it could be sufficient to detect cyber-attacks.

4.1 AI for intrusion detection

When we talk with a security expert about machine learning and behavioral detection, he would say it does not work and probably that it will never work. This feeling is quite reasonable in light of the original (and more current) research works.

Cyber-complexity The first reason explaining why AI failed in cyber-security while it revolutionizes other fields is the context specificity. Solving the GO game, recognizing human faces or making a car autonomous are real feats but detecting zero-day attacks is far harder. Designing algorithms to this purpose is really demanding and we can list some of the difficulties:

- All the networks are different, so adaptable techniques are paramount
- To detect new attacks, algorithms must not be stuck to training observations, so supervised methods are not suitable
- As the context is constantly evolving, dynamic models are preferred
- Online detection is required to prevent systems from being damaged as soon as possible

Historical background The pioneering work of Denning [23] is one of the most relevant in intrusion detection. She has given a model for a real-time IDS based on the detection of abnormal behaviours. Then several solutions, anomaly-based and also rule-based

(expert systems), have been proposed and some of them are described in [91]. After these initial works (until the late 1990s), automation has taken a great part in the research in order to exploit larger data volumes and make systems more efficient while reducing human intervention. Underlying models were now build from training data and not from expertise [126].

A large amount of work was done in the early 2000s. Actually, many surveys about ML/DM techniques for intrusion detection [9, 13, 38, 126] analyze publications from those years.

KDD99 performance race Several evidence can explain the increasing research interest in the early 2000s. Undoubtedly, the improvement of ML/DM algorithms (in concert with the computation power) is a key element. Moreover, the releases of specific datasets from the DARPA [77, 78] made the training, the evaluation and the comparison of anomaly-based intrusion detection techniques far easier.

In particular, KDD99 [120], a derivative of DARPA98 is currently the most widely used dataset to benchmark IDS through the whole literature. However, in spite of its practical aspect, KDD99 is a problematic dataset. From 2000, McHugh [87] was yet criticizing it, mainly pointing the procedure to generate the data as they were synthetic and not representing real world traffic. Then, the analysis of Mahoney and Chan [81] and Tavallaee *et al.* [122] followed, highlighting some simulation artefacts, the records redundancy and the lack of exact definition of attacks.

More than that, this dataset may encourage supervised methods as the observations are labeled. In fact, such approaches only generalize signature-based techniques making the known rules potentially more robust [33]. They logically cannot be used to build a pure anomaly detector.

However, the 2000s saw the application of all the common and well proven ML/DM methods on this dataset (ANN [10], HMM [61], Naïve Bayes [3], Random Forest [60], SVM [57] etc.). See for instance [97, 13] for a rich description of these techniques to cyber-security.

Their results cannot be denied nevertheless the deployment of such methods in real-world contexts remains uncertain. The need of labeled data for training, the use of computationally expensive algorithms and the required expertise for parametrization may break the demands from the cybersecurity field.

Improvements and new challenges Many recent publications still use KDD99 to test their (often supervised) algorithms [2, 72, 112]. However, several research works have gone beyond by developing unsupervised techniques [100, 73] and testing it in real world contexts [14, 27]. Unsupervised algorithms are very appealing because they do not require labeled data for learning. Nonetheless, the counterpart is that they have several parameters to tune so as to reduce their [often high] false alarm rate.

4.2 Examples of unsupervised anomaly detectors

In this section we present two anomaly-based IDS which seems, to our point of view, among the most relevant in the literature. These publications differentiate from other works insofar as they aim to build real-world IDS. Thus, several efforts are clearly made to tackle the aforementioned challenges. Even if the proposed algorithms also have some drawbacks, these works mark out the real progress in this field.

UNADA The Unsupervised Network Anomaly Detection Algorithm (UNADA) is an anomaly-based IDS published in 2011 by P. C. Hernandez, J. Mazel and P. Owezarski [14]. It has been reintroduced and improved in few other publications like [15, 26, 27]. Here we present the basics of the algorithm and the reasons why this publication is relevant for us.

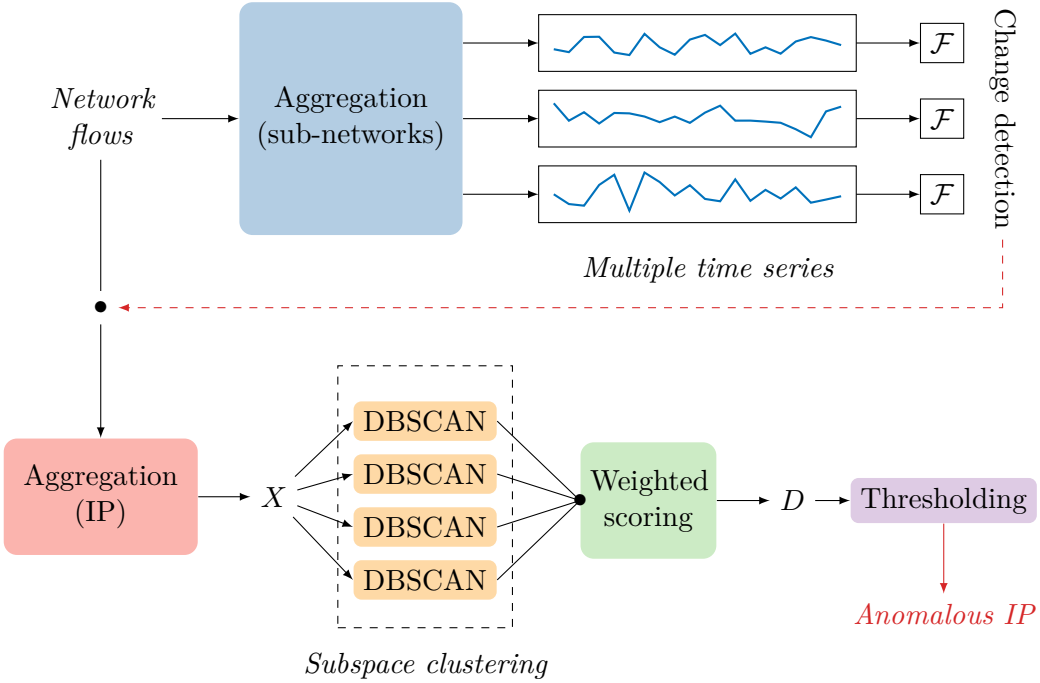


Figure 4.1: *Unsupervised Network Anomaly Detection Algorithm (UNADA)*

The figure 4.1 shows the pipeline of UNADA. In a word, UNADA performs anomaly detection in network flows through aggregated statistics monitoring and sub-space clustering. Initial data are basic network flows (like Cisco NetFlow[4]). For all the incoming flows, UNADA compute statistics (batch aggregation) at different network levels (source and destination with masks from /8 to /32). All these statistics are monitored along time (multiple time series) and an algorithm $\mathcal{F}$ detects some

[4] https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html

changes. When a change occurs, the initial flows are re-aggregated at IP level only (source and destination), building then a matrix X which contains n observations (number of different IP) of p features (aggregated statistics). Clustering algorithms are applied in all 2-dimensional sub-spaces. Each of them computes a score for all the observations. For all the observations, the sub-spaces clustering scores are weighted to finally output a final score. Eventually, a threshold is used to decide which observations are really anomalous among the initial data slots.

This publication is relevant for us for two reasons: the algorithm is totally unsupervised and there is a good trade-off between the amount of ML/DM techniques and the expertise. Indeed, the authors of UNADA detail the features they use and the reasons why they are relevant to detect cyber-attacks. These information are very valued in the cyber-security field. Besides, this expertise makes UNADA rather interpretable since an abnormal events is basically an IP address and we can post-analyze the features involved (according to the results of the sub-space clustering). Therefore, we can even classify the type of the attack according to these features. Also we have to note that the authors do not use KDD99 to test their algorithm but rather use real-world data from the MAWI platform [36]. In the ML/DM point of view, the whole approach is both logical and cumbersome. Every steps are coherent but the pipeline requests many operations: aggregations at many levels, anomaly detection for all the features at all aggregation levels and sub-spaces clustering. Its implementation is then not easy as it needs to link several different blocks. Besides some points are not clarified: what change detection algorithm $\mathcal{F}$ to use? Such an algorithm is also very likely to request parameters. What threshold to use in the end of the pipeline? As a static threshold it also suffers from all the points we have mentioned in section 1.3. Authors suggest to use an *elbow* method on the scores D but the way to compute such a threshold is not given. What parameter values should we use in the DBSCAN runs? It commonly needs to be tuned to avoid aberrant results. Below we sum up some key elements about the UNADA algorithm.

Pros	Cons
• Totally unsupervised	• Heavy pipeline
• Simple and meaningful features	• Many parameters
• Can classify anomalies (taxonomy)	• Not real-time ($O(n \log n)$ complexity)
• Parallelizable	• Hardset thresholds

Kitsune Kitsune [90] is a more recent network intrusion detection system (NIDS) published in 2018 by Y. Mirsky, T. Doitshman, Y. Elovici and A. Shabtai. As of today, it can be considered as the state of the art among the anomaly-based NIDS, insofar as it uses recent ML/DM primitives and their experimental results seem the most convincing for a real-world use.

Kitsune mainly uses auto-encoders (AE) to perform anomaly detection. Thus, they need to be trained before any execution. This phase is described on figure 4.2. Initial data are packets, considered as normal observations, which are aggregated so as to

produce statistics (this step is called *feature extractor*). The aggregation keys are let to the users but the authors give some examples: IP source (mean and standard deviation of the packet's size), IP tuple or port tuples (covariance of the packet's size between both direction) etc. The aggregation is not performed in a common batch way but incrementally with a damp effect. It notably reduces the computation and the memory during the execution stage. However, a decay factor λ must be set (analogous as a window width of a batch aggregation). In [90], authors use also several values of λ and then multiply the number of statistics. In the end, they need to manipulate about $P \simeq 100$ statistics, which describe the "current" traffic behavior.

During the incremental computation of the statistics, correlation between them are also updated (we note Σ). This correlation is used as a distance between the P features during a hierarchical clustering stage. The cut is performed the highest possible so that the clusters gather at most m features (a user-defined parameter). These bags of features $F_1, F_2 \dots F_k$ are actually computed so as to reduce the complexity of the next step. So, this clustering stage merely builds a *feature mapper* which maps the initial statistics vector S to k sub-vectors $S_1, S_2 \dots S_k$.

Now, let us describe the final *anomaly detector*. Every mapped statistics S_i is used to train a 3-layers auto-encoder. Its input/output layers contain naturally $|F_i|$ neurons, the number of features in F_i. The hidden layer has $\lceil \beta |F_i| \rceil$ neurons where $\beta \in [0, 1)$ is a user-defined parameter (authors use $\beta = 3/4$ in their experiments). The k auto-encoders each output a reconstruction error (RMSE) which is used to feed and train another 3-layers auto-encoder with k input neurons, $\lceil \beta k \rceil$ hidden neurons and k output neurons.

The execution phase (figure 4.3) mainly re-uses the parts of the training phase except that auto-encoders are executed instead of trained. At every incoming packets, a new statistics row S is computed (damped incremental statistics). This row is split into sub-rows $S_1, S_2, \dots S_k$ according to the feature mapper previously learned. These sub-rows are fed to the anomaly detector which outputs a score from the last auto-encoder, which is its reconstruction error. Authors suggest two thresholding methods: either a hard-set threshold ϕ to tune (they uses it to compute AUC in the experiments), or a quantile assuming the distribution of the scores as log-normal.

This contribution is interesting as it proposes a real-time algorithm based on "modern" anomaly detection primitives (auto-encoders). As it works at network packet level, anomalies are precisely related to an atomic network event. Regardless of the whole pipeline, this publication presents a complete experimental setup following some real-world scenarios (Reconnaissance, Man in the middle attacks, Botnet attack etc.) in various contexts (IoT network and surveillance network). In addition, all the capture files are publicly available[5]. This testing stage really breaks with what has hitherto been made. Eventually, authors highlight very good results in comparison with signature-based IDS (Suricata[6]) and other anomaly detectors.

[5] See `https://github.com/ymirsky/Kitsune-py` for more details

[6] `https://suricata-ids.org/`

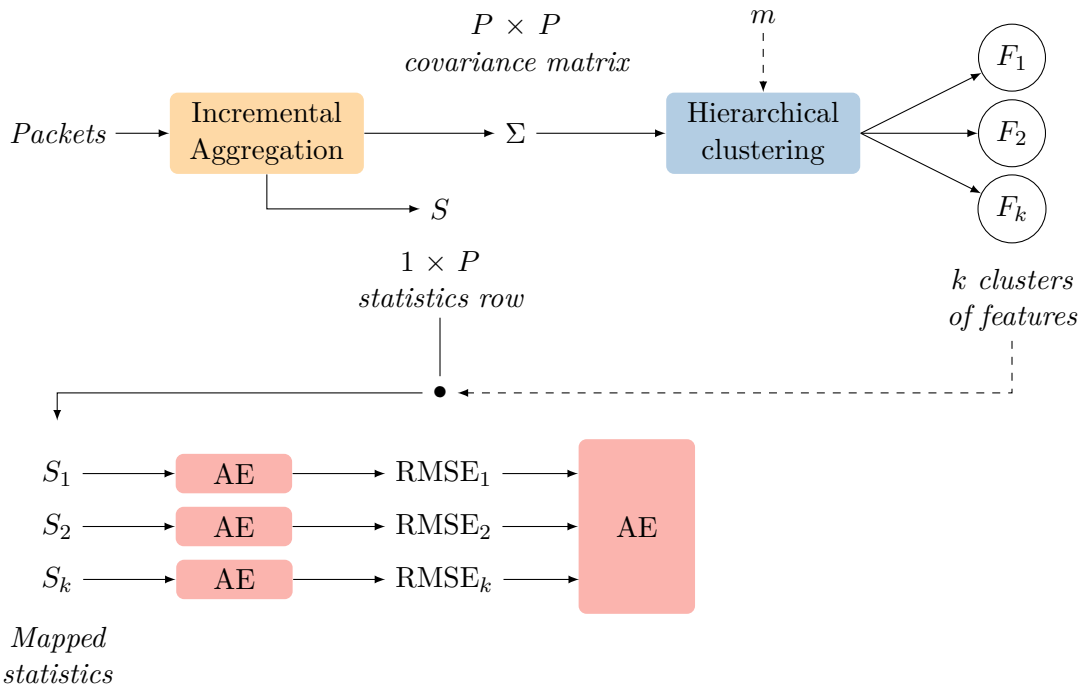


Figure 4.2: *Kitsune algorithm during the training phase*

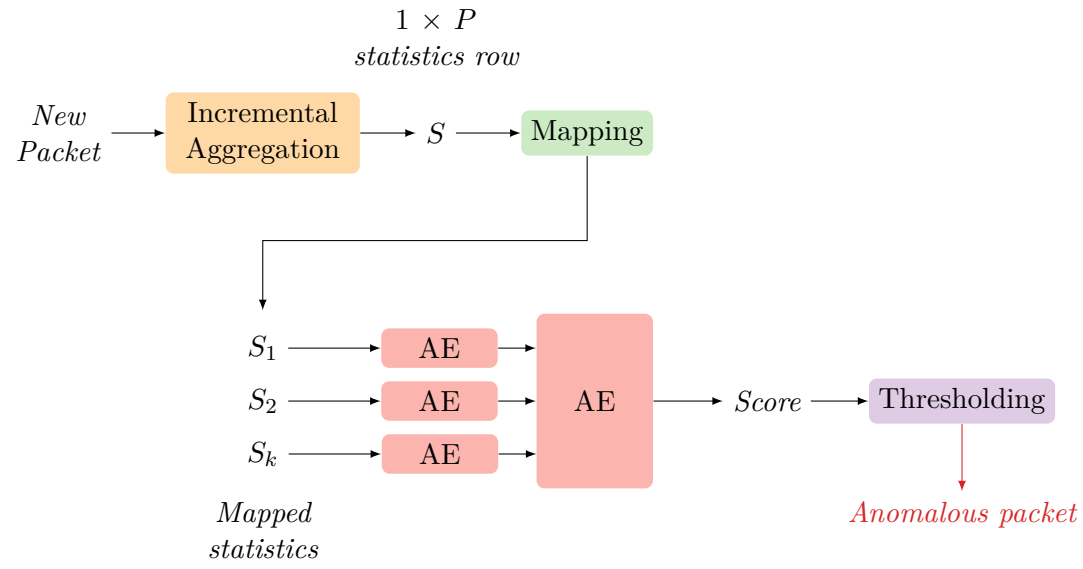


Figure 4.3: *Kitsune algorithm during the execution phase*

However Kitsune is not free from all reproaches. First of all, data are assumed as normal during the training phase (we could then retort its unsupervised character).

This assumption is very strong for the cyber-security community. In practice, we cannot ensure a clean network at any moment. The issue is that this is a key assumption in the Kitsune design and we have no clear idea about its behavior in case of data contamination during the training step.

Furthermore, unlike UNADA, Kitsune learns once and then executes without updating its knowledge (UNADA flags behavior changes). Actually, once the AE are learned, the feature mapper cannot change otherwise AE must be re-trained (with normal data). Kitsune is then quite static and that may be a real problem if we monitor a constantly evolving traffic.

Kitsune uses the power of auto-encoders but it also suffers from their lack of interpretability. When a packet is declared as abnormal, it is hard to understand the reason: in particular we cannot easily get back the involved sub-statistics S_i.

The pipeline can also be difficult to understand, not as clear as UNADA one: using the RMSE of initial auto-encoders to feed another AE is rather uncommon. The goal of the authors was to avoid a single "high" auto-encoder which is expensive to train. Besides, some of their best results have been made with $m = 1$ meaning that an AE is used to monitor every feature. In this context, all these AE basically contain 3 neurons: one input, one hidden and one output. We can naturally question the relevance of this architecture.

Like UNADA, many choices are let to the users which does not ease its setup: statistics to compute, values of λ (decay factor), value of β (AE compression rate), parameter m (maximum height of an AE) and the decision threshold ϕ. Below are summarized the characteristics of Kitsune.

Pros	Cons
• Modern primitive	• Benign training set assumption
• Real-time detection	• Learns once and execute (static)
• Packet-level anomalies	• Lack of interpretability
• Extensive tests	• Several parameters (questionable design)

4.3 NetSpot: a simple IDS with statistical learning

Henceforth, we present the Network Intrusion Detection System (NIDS) we have developed around the SPOT algorithm, called `netspot`. In this section, we try to take the *software engineer* point of view, meaning that there is no algorithmic considerations but we rather detail the operation and the features of `netspot`.

Motivations When we decide to implement such a software, the initial motivation was only to show that we can build a basic IDS powered by SPOT, namely a proof-of-a-concept. The aim was neither to present better results than other ML/DM based IDS, nor to concurrent other IDS solutions (signature-based). Our goal was to build a very simple IDS but with a statistical learning core (SPOT). Thus, `netspot` should be

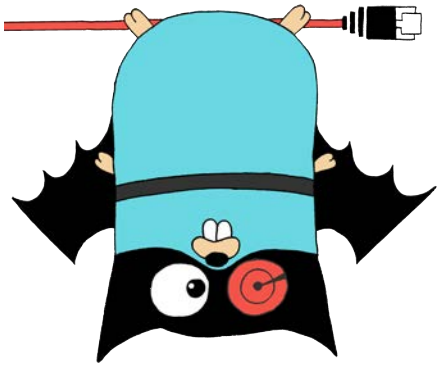


Figure 4.4: *Logo of* `netspot`

viewed as a component of a more general monitoring system, able to provide a relevant behavioral information about the network traffic.

About the implementation `netspot` was initially in `python` but quickly re-coded in `Go`, mainly for performance and concurrency reasons.

Design In a word, `netspot` monitors network statistics with the SPOT algorithm. At the basis, `netspot` increments some network counters once a packet is received. This operation is lightweight as the counters rather perform "atomic" operations (e.g. looking at flag, retrieving an IP address, computing the packet size etc.) This part of the program is called the MINER and mainly uses the `GoPacket`[7] library, which wraps around [the famous] `libpcap`[8], to parse network packets.

The figure 4.5 illustrates the MINER's task. When a new packet is received, the MINER extracts some layers if they exist: Ethernet, IP, ICMP, TCP and UDP. Every layer will be useful only for specific counters, so the MINER dispatches them according to the counter needs. For example, the `IP` counter can only receive IP layers and it increments an internal accumulator every time it receives such a layer. But the `SYN` counter, which accepts only TCP layers, increments itself once the received TCP layer has a SYN flag set to 1.

Above the MINER, we built the ANALYZER which manages the network statistics. The statistics merely need counters to make their computation. For instance, the `R_SYN` statistics (ratio of SYN packets) divides the `SYN` counter by the `IP` counter. At given interval (e.g. every 5 seconds), the ANALYZER asks the MINER the counter values and compute the statistics. This statistics corresponds then to the ratio of SYN packets the last 5 seconds (the ANALYZER also asks the MINER to reset its counters).

The statistics are not as basic as the counters (i.e. elements making a single computation) because they embed a SPOT instance (or more generally a DSPOT instance) which monitors what they compute in real time. Finally the ANALYZER has three outputs: the raw statistics, the SPOT thresholds and some alerts (see figure 4.6).

[7] `https://github.com/google/gopacket`
[8] `https://www.tcpdump.org/`

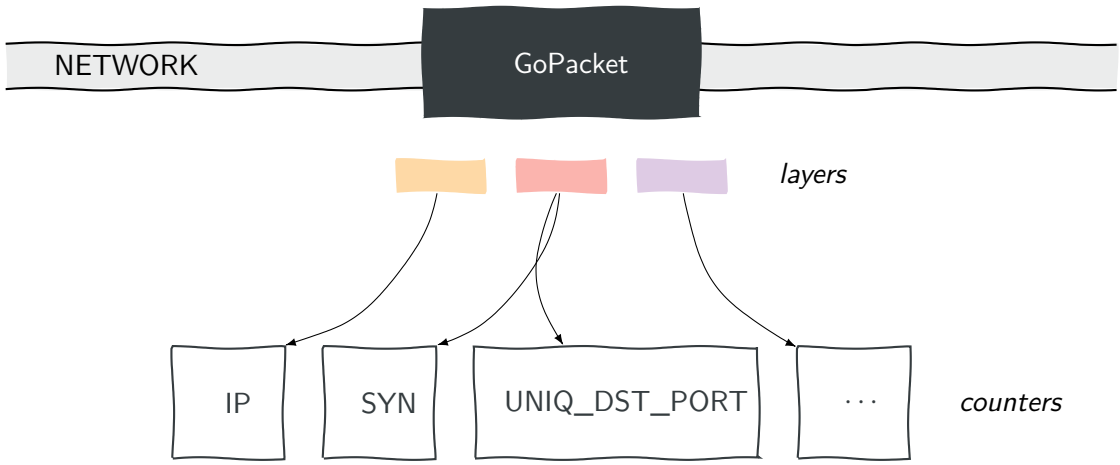


Figure 4.5: *The* MINER *component. It parses network packets, extracting layers and dispatching them so as to increment the counters*

Command & Control `netspot` runs as a server (through a `systemd`[9] service or a `docker`[10] container for instance) and can be managed by clients. The server can be configured through a simple file or through command line arguments. In particular, some statistics can be loaded and their SPOT instances can be individually parameterized.

We also develop a `Go` client (command line interface), called `netspotctl`, which currently uses the RPC facilities provided by `Go` to interact with the server. However, `netspot` also exposes a REST API, with an OpenAPI[11] specification, making the implementation of other HTTP clients rather easy and even "automatizable". At the client level, simple operations are available and the table 4.1 details what the API allows.

Integration Currently, `netspot` outputs three JSON files, described in the figure 4.6: the raw statistics, the thresholds computed by SPOT (one for each statistic) and the alarms (when a value is beyond a threshold). However `netspot` is also able to send these information to an `influxdb`[12] database, which is "a time series database designed to handle high write and query loads". Such a storage eases the information aggregation. For instance Security Information and Event Management softwares (SIEM) are likely

[9] `https://freedesktop.org/wiki/Software/systemd/`
[10] `https://www.docker.com/`
[11] `https://www.openapis.org/`
[12] `https://www.influxdata.com/products/influxdb-overview/`

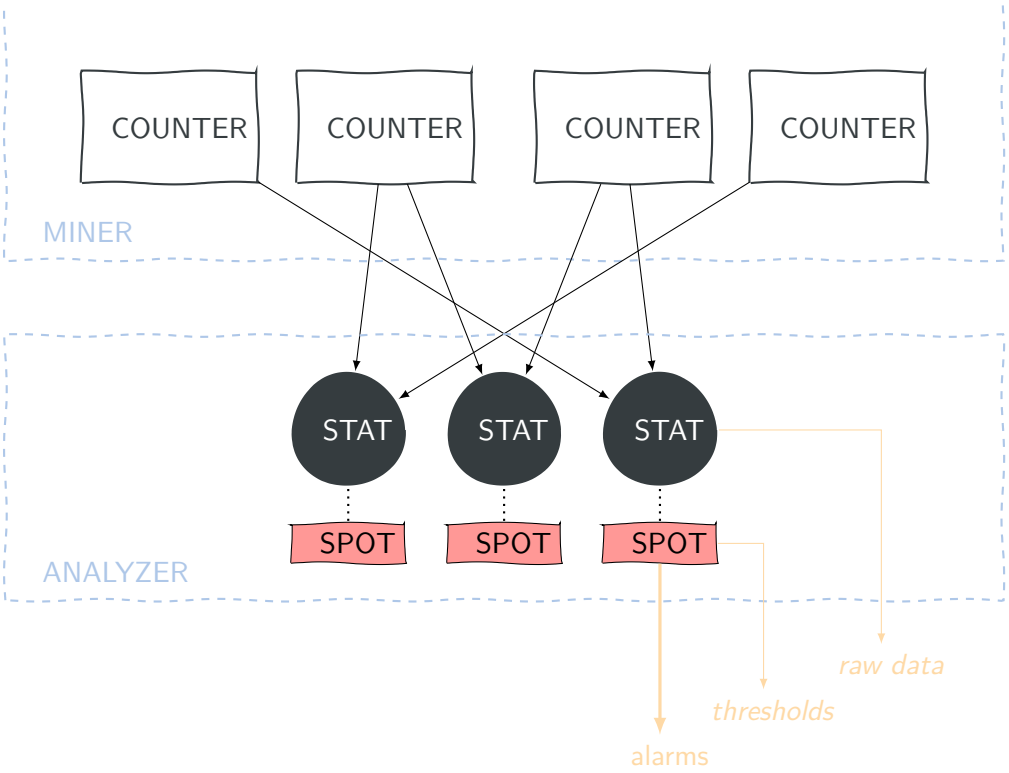


Figure 4.6: *The* ANALYZER*: statistics request periodically counter values, compute stats and feed them to an embedded SPOT instance, which can trigger alerts*

to handle real-time data from multiple heterogeneous sources. Using such a technology to store events makes `netspot` easier to integrate in a general monitoring system. At a first level, one can use a visualization technology upon `influxdb`, like `Grafana`[13], to build a dashboard showing the `netspot` events (see figure 4.7).

Modularity `netspot` can also be adapted to specific needs. Indeed, the main abstractions behind `netspot` are the counters and the statistics. A developer can easily implement its own counters and then its own statistics. Thus, one may imagine the need to monitor the activity of some particular IP addresses or sub-networks (like a server or the external network). In this example, new counters and new statistics (written in `Go`) should be added to the sources and `netspot` must be re-compiled. Especially, we provide a simple process for the developer to enrich `netspot` as he sees fit.

[13] `https://grafana.com/`

Action	Comment
Set the device to monitor	It could be either a network capture file or a network interface
Load/Unload statistics	The `netspot` server automatically loads or unloads the counters according to the statistics
Set the computation period	This is a key parameter which must be adapted to the traffic `netspot` monitors
Start/Stop/Reset the monitoring	The Reset action merely deletes the internal state of the server while Stop rather means a pause
Other MINER configurations	Examples: timeout, snapshot length and promicuous mode
Other ANALYZER configurations	Examples: output directory, database connection
Introspection & misc.	Examples: status of a statistic, list of available/loaded statistics, list of available interfaces...

Table 4.1: *List of the main available actions from a* `netspot` *client*

Figure 4.7: *Example of a* `Grafana` *dashboard to visualize the* `netspot` *output*

Discussion The task of `netspot` seems very basic: it independently monitors network statistics and flags abnormal behaviors. Actually `netspot` takes the opposite view of what has been previously done: no heavy pipeline, no expensive ML/DM stage and

no laborious configuration. Only the statistics computation period and the (D)SPOT instance need to be set (mainly the parameter q of SPOT). The former can easily be defined by a security expert (according to the network where `netspot` is deployed) while the latter does not require a fine-tuning stage. In the next section, we will observe that such a design is enough to detect cyber-attacks.

So what are the advantages of other methods? Actually, they provide a fine-grained anomaly detection resolution (at flow or packet levels) while `netspot` flags small time windows. A deeper investigation would then be necessary to catch, for instance, the IP addresses of the attacking systems. In fact, this step is embedded in Kitsune and UNADA so the relevance of `netspot` might be questioned. However, we think that we do not need full built-in solutions but rather heterogeneous "atomic" sources of information, collected, aggregated and correlated in back-end. `netspot` can be such a source and we have made many efforts to make it really practical (efficient code with a modular and integrable design).

4.4 NetSpot facing real-world attacks

In 4.2, we mentioned the relevance of the experiments made by the authors of Kitsune [90]. In particular they have reproduced some common attack scenarios, providing the corresponding network captures[14]. In this section, we test `netpot` on some of these scenarios. Our results justify the "simplicity" of `netspot`: monitoring single statistics is enough to detect abnormal events. At the end, we deal with practical performances on two different systems.

SYN DoS In this first scenario, an attacker "disable a camera's video stream by overloading its web server". This is a classical deny-of-service (DoS) attack which flood a system by sending a huge amount of SYN packets. The ratio of SYN packets (number of TCP packets with SYN flag over the number of IP packets) is then a natural statistic to catch such attack. The figure 4.8 shows the monitoring by `netspot`. One may observe that the threshold learned (dashed line) on the first value allows to flag abnormal events occurring in the end (red spots).

Besides, this attack also impacts the ratio of ACK packets. When a user sends a SYN packet, the server responds with SYN-ACK packet and wait for a last ACK packet from the user (3-way TCP handshake). In this scenario, the attacker sends the first SYN packet but he ensures that the last ACK packet will not be sent (spoofed or rogue IP can be used). So the server wait for the ACK packet. If many SYN packets have been sent, the server will use all its resources waiting for ACK packets. A legitimate user who sends a new SYN packet will not have the SYN-ACK reply (denial-of-service). At this moment, the number of ACK packets intuitively falls.

[14] data are available at `https://drive.google.com/drive/folders/1kmoWY4poGWfmmVSdSu-r_3Vo84Tu4PyE`

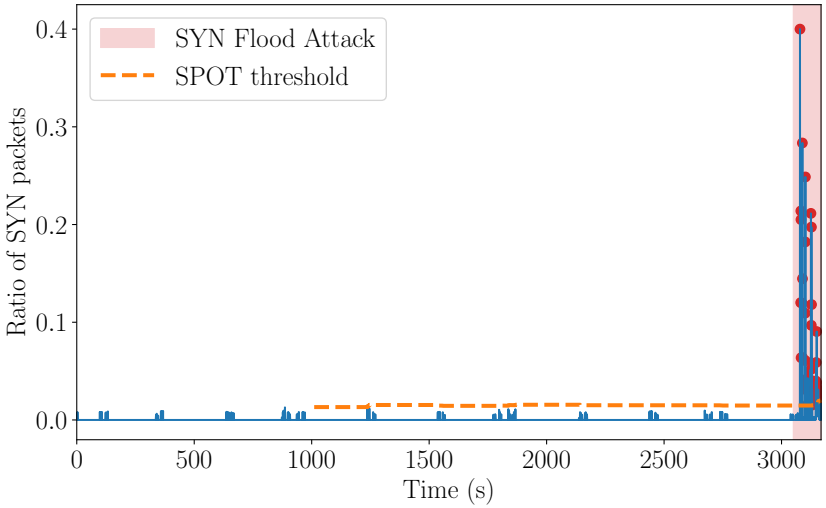


Figure 4.8: *Monitoring of the ratio of SYN packets (500ms time window) to detect DoS attack*

The figure 4.9 particularly shows this phenomenon during the attack. By computing a lower threshold (dashed line), `netspot` can easily flags the fall of the ratio of ACK packets.

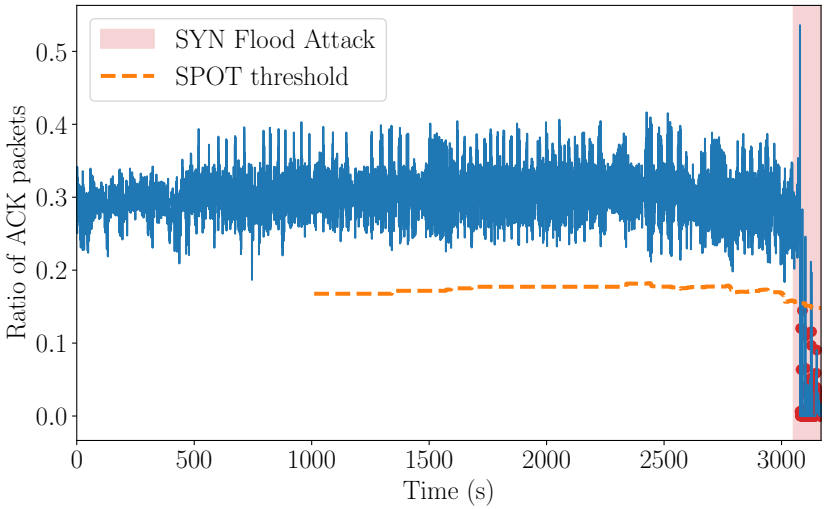


Figure 4.9: *Monitoring of the ratio of ACK packets (500ms time window) to detect DoS attack*

Mirai Mirai is a malware that aims to infect Linux devices (notably IoT devices). Once several systems are infected, the botnet thus created can be used to launch Distributed Denial-of-Service attacks (DDoS). Among highly publicized Mirai attacks, one can mention the attack on OVH[15] (French web host) in September 2016 and the attack on Dyn[16] (DNS provider) in October of the same year.

Once Mirai has infected a system, it tries to infect others. For that purpose, it scans the network so as to discover potential targets. A simple solution to scan the neighborhood is to broadcast ARP (Address Resolution Protocol) probe packets and wait for responses. Indeed, ARP is basically used to find MAC address associated to a given IP address. When a host receives an ARP probe, it sends back its IP and MAC addresses.

The experiment proposed in [90] considers an IoT Wi-Fi network (3 PC and 9 IoT devices) with a security camera infected by a real sample of the Mirai malware. The corresponding network capture notably records this attack discovery stage. The figure 4.10 shows the detection by `netspot` of the Mirai scan.

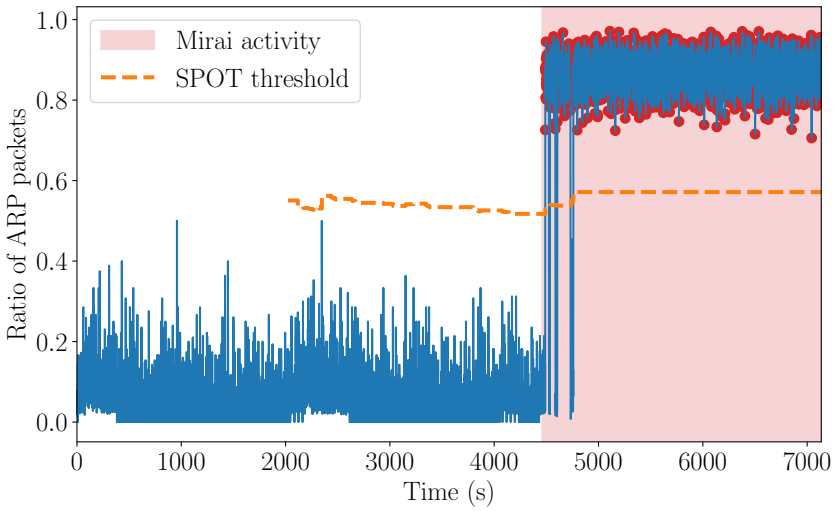


Figure 4.10: *Monitoring of the ratio of ARP packets (1s time window) to detect Mirai activity*

Active Wiretap In this last scenario, an attacker "intercepts all LAN traffic via active wiretap (network bridge) covertly installed on an exposed cable". Practically, a Raspberry Pi (in promiscuous mode) is plugged on a switch and eavesdrops the local network. To detect such an attack, we may wonder what happens when an Ethernet

[15] `https://www.ovh.com/world/`
[16] `https://dyn.com/`

cable is plugged. Naturally, it also starts with a discovery phase where the new system asks the LAN who is connected at some IP addresses. Possibly other requests can raise according to the system (network time, name resolution etc.).

Like Mirai, many ARP packets are likely to be broadcast on the network. On figure 4.11, we can observe the behavior of the ratio of ARP packets. This rate is rather low but two high peaks are noticeable. If we investigate deeper, these peaks are not the most interesting. However, the zoom on the plot (bottom figure) highlights small peaks flagged by SPOT (red circles). The latter are especially created by the Raspberry Pi activity (the wiretap). It means that `netspot` is able to detect low amplitude anomalies which obviously is a paramount feature for an IDS.

Implementation and performances Keeping in mind the practical stakes, we propose several ways to get `netspot`. First of all, code of `netspot` is available on Github[17]. Moreover, binaries of the current version (1.3) can be downloaded through a `debian` package or a `docker` image (both are available for `amd64` and `armhf` architectures).

The simplicity of `netspot` makes it fast. To check its performances, we feed a big capture file (2 278 689 packets) to `netspot`, varying the number of statistics to monitor. When `netspot` processes the file, the current packet parsing rate can be retrieved. We have noticed that `netspot` is not slower once it is calibrated, so the performances take all into account: the cold start (calibration) and the cruising regime (flag/update). The figure 4.12 shows the performances of `netspot` on two systems: a classical desktop computer (8 cores Intel i5-8250U and 16GB RAM) running `KUbuntu`, and a Raspberry Pi 3B+ (4 cores ARMv7 and 1GB RAM) running `Raspbian`.

On the desktop, `netspot` is roughly able to process 500 000 packets a second (with some peaks higher than 1M packets/s). On the Raspberry, `netspot` is about ten times slower with about 50 000 packets/s. To compare with Kitsune [90], the latter is able to process about 40 000 packets/s on a desktop computer (8 logical cores Intel i7-4790 and 4GB RAM) and about 5 000 packets/s on a Raspberry Pi. On rather equivalent configurations `netspot` is therefore 10× faster than Kitsune. To compare with a signature-base IDS, we also run `Suricata` on the same capture file with default rules[18] (20434 signatures). On the desktop computer (there is no release of `Suricata` targeting a Raspberry Pi), `Suricata` analyzes the whole capture (≃2M packets) in about 5s, so it leads to an average packet rate of 400 000 packets/s.

4.5 Conclusion

Applying ML/DM techniques to intrusion detection is a very hard task since goals are not clear and relevant data are lacking. While the literature on this field is substantial, the presented results are generally not usable in practice. Several publications try to circumvent the hindrances in providing more practical algorithms. They do not need

[17] `https://github.com/asiffer/netspot`

[18] A `python` tool allows to update the rules, see `https://github.com/OISF/suricata-update`

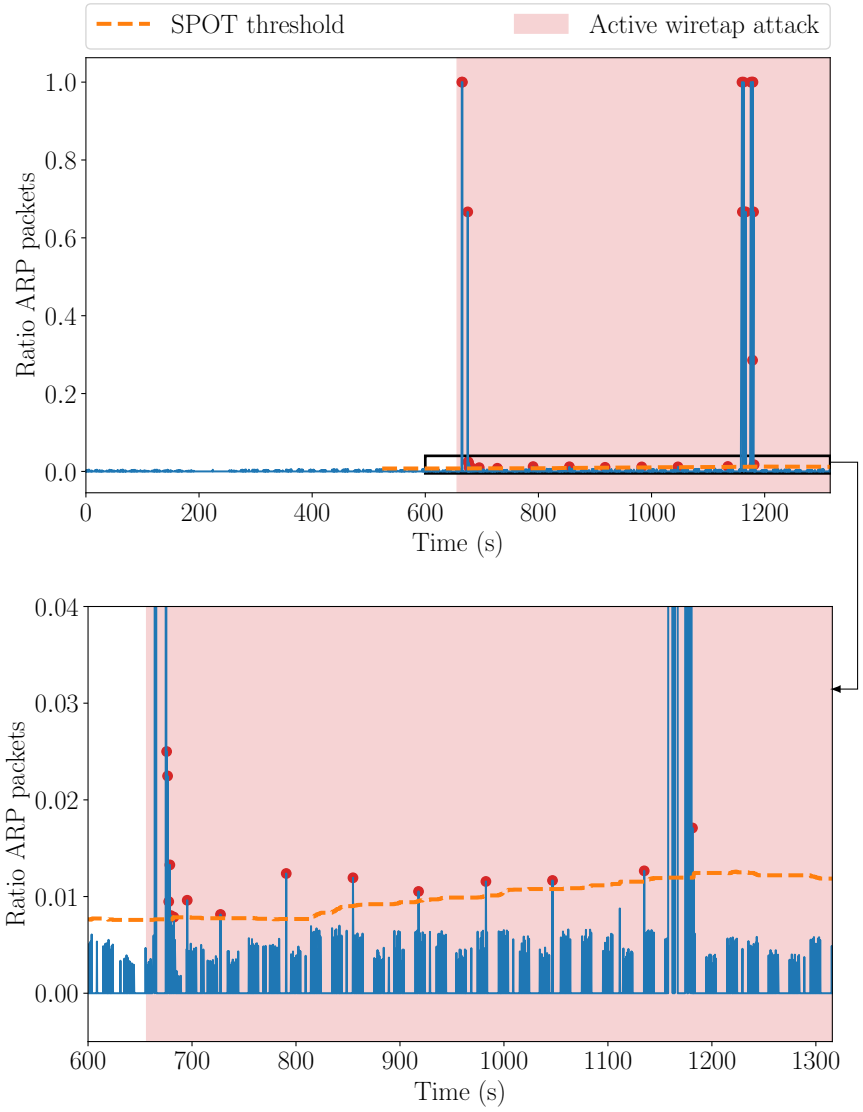


Figure 4.11: *Monitoring of the ratio of ARP packets (250ms time window) to detect active wiretap*

labeled data (unsupervised), they work online and they are efficient on real-world scenarios. Unfortunately, they are quite complex (heavy pipeline, many parameters to tune) because they aim to build end-to-end solutions (no integration in a more general monitoring system).

`netspot` is a new simple IDS embedding statistical learning. This is obviously not the ultimate software solution to detect all the zero-day attacks, but merely a proof-of-concept showing that a simple design can already provide relevant information. Indeed,

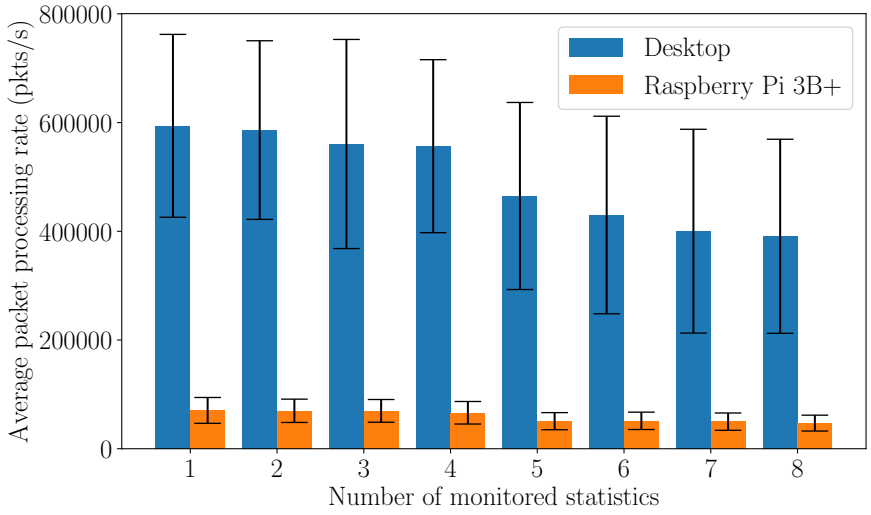


Figure 4.12: *Performances of* `netspot` *on two systems according to the number of monitored statistics*

though some experiments we have shown that `netspot` is fully capable of detecting real-world cyber-attacks. Moreover we have implemented `netspot` with the desire for practicality in mind. In reality, `netspot` is simple, fast, modular and easy to integrate in a more general monitoring system.

`netspot` is still under active development as possible improvements are plenty. We hope that it could be a modest but credible example of practical anomaly-based IDS.

Part II
Unimodality testing

Given a multidimensional dataset of numerical attributes, an important question is to understand the "grouping behavior" of the data points. This question is traditionally answered by *clustering algorithms*. "Grouping behavior" being an ill-defined concept, clustering algorithms answer a more precise question : they determine groups (clusters) of data points that are similar, while being different from the data points of the other groups. From the statistical point of view, findings clusters means finding high probability areas. If we imagine that the observations are generated through a given distribution it would mean that clustering is looking for the *modes* of the distribution, basically the *bumps*.

Clustering can be an expensive task and many algorithms are not able to scale with the number of observations (Mean Shift [20] has at least $O(n \log n)$ complexity and Affinity Propagation [37] is quadratic in the number of points for examples). Obviously we may also cite $k-$means [59] or DBSCAN [31] which are linear in n but the first needs the number of clusters as input and the second may output very different results according to its parameter values. Sometimes, one may not need such a detailed result: a more basic "grouping behavior" question can be to determine if all the data points make one single, coherent group or not.

In statistical terms, this is formulated as determining if the distribution of the data points is *unimodal* or not. *Per se*, unimodality test is a first information about the data, as it tells that the data points cannot be obviously separated into distinct groups. In a medical experiment, this would for example tell that the patients all reacted in a similar way, and that no group of patients deviated significantly from the others. Unimodality can then be seen as a building block for clustering algorithms: first, it can be used as a low-cost test to determine if running a clustering algorithm is necessary or not. Second, it can be used to improve clustering results, for example to help in parameter estimation.

Unimodality tests exist in the literature and have mainly been proposed in the 1980s and 1990s. The main problem of these tests is that they are unable to tackle multi-dimensional data. Besides, they generally rely on optimization problems which make them unable to work with streaming data.

Our main contribution is to propose the first statistical unimodality test designed for multidimensional data, called Folding Test of Unimodality (FTU). Furthermore, this test is the first able to deal with streaming data. Our approach relies on a novel descriptive statistics for (multidimensional) data, that provides a measure of its unimodality character (*level of unimodality*). The proposed approach keeps the same elegant properties as well-known unimodality tests like *dip*: it is non-parametric, independent from the distribution of the data and it uses only a $p-$value threshold to tune the confidence in the results.

FTU, in the univariate case, relies on the interplay between symmetry and variance. This idea proved to be very powerful. However, the generalization to the multivariate case introduces some limits and drawbacks. The main limitation is that the underlying optimization problem may not have a solution. This requires to use approximations, which may cause FTU to fail when facing certain multimodal symmetrical distributions. To circumvent these drawbacks we also propose the

Hyperplane Folding Test of Unimodality (HFTU) which is another generalization of FTU to higher dimensions. HFTU provides more theoretical guarantees making it more robust than FTU against data distribution. In addition, HFTU builds an hyperplane $\mathcal{H}^*$ which provides a global description of the distribution underlying data.

All these features are real assets to build clustering algorithm. In this purpose, we also propose $\Phi-$means, a clustering algorithm which incrementally finds the k of $k-$means. Unlike other common $k-$means wrappers, $\Phi-$means is based on the *unimodality assumption of the clusters* (UAC) and is then more general than common gaussian-based clustering algorithms while keeping similar performances (linear dependency in the number of observations). Precisely, it uses HFTU to estimate the validity of the clusters.

First, we will detail the related work on unimodality testing. Then we will present the core idea of our new test, called *folding mechanism*. The latter will be at the basis of both the Folding Test of Unimodality (FTU) and the Hyperplane Folding Test of Unimodality (HFTU). Finally, we will more focus on the application of unimodality testing to clustering through the $\Phi-$means algorithm.

5

Previous unimodality tests

One may view and then define unimodality in different ways. Such point of view naturally has an impact on how to check this characteristic on a given distribution. A common way is to look at the *bumps* (the local maxima, or *modes*): a unimodal density has a single bump although a multimodal density have several ones. One may also uses the fact that a unimodal cdf is first convex and then concave while a multimodal cdf alternates several times between these two states. These univariate definitions are actually special cases of more general definitions of unimodality (given for multivariate distributions). They actually all coincide in the univariate case while they remain non equivalent in the general case [24]. Thus, the concept of unimodality for univariate distribution is clear nonetheless it must be precised in the multivariate context.

Many statistical tests to assess unimodality (or multimodality) have been proposed in the literature and Stoepker gave recently a nice survey of them [119] (one may also have a look to the thesis of Minotte [88] for earlier contributions). The approaches commonly test unimodality versus bimodality but they can be adapted to test unimodality versus k modes. The great majority of the previous tests deal with univariate distributions but an example of multivariate test will also be detailed.

Despite their theoretical soundness they have some drawbacks like the need to estimate the distribution, the complexity of implementation and the computational

cost. These shortcomings limit their adaptation to more general contexts, namely high dimensional and/or streaming data.

5.1 The Silverman's test (1981)

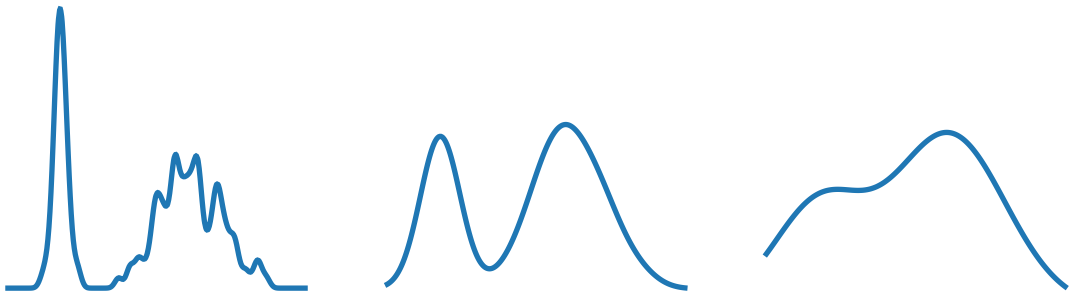
The Silverman's test [117] is a univariate unimodality test which estimates the density with a gaussian kernel. This test is still widely used in many domains including biology [127], ecology [7] and economy [110]. Particularly, an R package is available[1] but not on the official repository (CRAN).

Kernel estimator The Silverman's test [117] is built upon a kernel density estimator $\hat{f}$ of the underlying density f. It tests the hypothesis "f has m modes" (H_0) versus "f has more than m modes" (H_1). Given n observations $X_1, X_2, \dots X_n$, the kernel estimator is given by

$$\hat{f}(t,h) = \frac{1}{n\,h} \sum_{i=1}^{n} K\left(\frac{t - X_i}{h}\right)$$

where K is the gaussian kernel (this choice brings both computational and theoretical advantages).

The parameter h is the window width which tunes the contribution of every observation X_i (it corresponds to the variance of every gaussian density). Consequently, a large h produces a smooth estimator and tends to make the modes disappear. Conversely the estimator is likely to shape a mode on every single observation when h decreases. An example on iris dataset (petal length feature only) is given on figure 5.1.



(a) $h = 0.1$ **(b)** $h = 0.5$ **(c)** $h = 1.2$

Figure 5.1: *Kernel estimator according to the bandwidth h on iris dataset*

We can notice than the larger h is, the smoother the kernel estimator is. This behavior is actually proven by Silverman in [117] (see theorem 5.1).

[1] `https://github.com/jenzopr/silvermantest`

THEOREM 5.1 (Silverman) *Let $X_1, X_2, \dots X_n$ be n observations and K the gaussian kernel. For every integer $m \geq 0$, the number of local maxima of $t \mapsto \dfrac{\partial^m \hat{f}}{\partial t^m}(t,h)$ is a decreasing function of h.*

Besides, Silverman defines the *critical window* h_{crit} as

$$h_{\text{crit}} = \inf \left\{ h \;\middle|\; t \mapsto \hat{f}(t,h) \text{ has at most } k \text{ modes} \right\},$$

thus $t \mapsto \hat{f}(t,h)$ has more than k modes when $h < h_{\text{crit}}$. The Silverman's test uses the fact that large values of h_{crit} are likely to reject H_0. In practice, the author claims that h_{crit} can be found by a binary search procedure but no automatic way to compute the number of modes is exhibited.

Statistical significance Let g be the true density from which $X_1, \dots X_n$ are drawn, and h_0 the computed critical window. If we suppose the null hypothesis, then the probability to get h_0 lower than the true critical bandwidth h_{crit} is then the probability to $t \mapsto \hat{f}(t,h_0)$ to have more than k modes, by definition of h_{crit}:

$$\mathbb{P}(h_{\text{crit}} > h_0) = \mathbb{P}\left(t \mapsto \hat{f}(t,h_0) \text{ has more than } k \text{ modes} \;\middle|\; \{X_1, \dots X_n\} \sim g \right).$$

Therefore, to compute the significance of the test, we merely need to draw samples from g and compute the number of modes of $t \mapsto \hat{f}(t,h_0)$ at each replication. In general, we cannot generate new samples from g so Silverman suggests to use the smoothed bootstrap procedure of Efron [29] to simulate g by sampling with replacement from the initial set of observations.

Improvements Several contributions discuss the Silverman's test. Particularly, some theoretical properties of the Silverman's kernel estimator are developed in [82, 118] for bounded densities. Hall and York [48] tackle the lack of accuracy on the test significance (induced by the bootstrap procedure). In the context on testing unimodality versus multimodality, they provide a more accurate procedure to compute the decision level λ_α given the significance α. The bandwidth h is a critical point of the test. As all the points embed the same gaussian mass (same h), it is not adapted to modes of varying sized. Minotte [89] notably proposes the mode-existence test which individually assigns a bandwidth to every observation.

5.2 The Excess Mass test (1991)

Unlike the majority of unimodality test, the Excess Mass Test of Müller and Sawitzki [92] tries to estimate both the location and the number of modes, which is actually stronger than testing unimodality. It is based on the excess mass functional which is likely to bring out the density bumps. Even if the test is theoretically designed to tackle multivariate densities, general effective algorithms are not available [109].

Excess mass functional The excess mass test of Müller *et al.* [92] mainly tests unimodality (H_0) versus bimodality (H_1). This test is based on the *excess mass functional* defined by

$$\mathbf{E}_f : \lambda \mapsto \int (f(x) - \lambda)^+$$

where $(x)^+$ is equal to x when $x \geq 0$, 0 otherwise. The figure 5.2 gives an illustration of the meaning of $\mathbf{E}_f$: it corresponds to the "excess" area of f about the level λ. In particular, we may understand how it may catch the modes, especially when λ increases. When the density f has exactly m modes, the functional can be expressed as

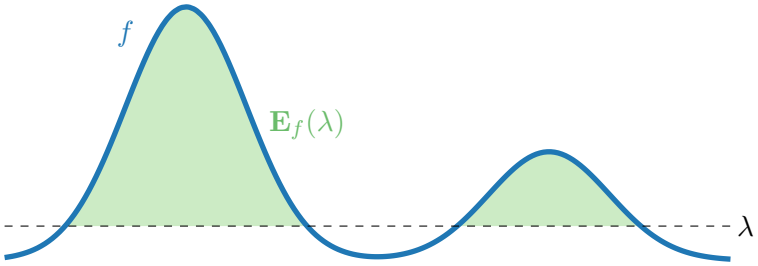


Figure 5.2: *Excess mass functional on a bimodal density*

$$\mathbf{E}_f(\lambda) = \sup_{\mathcal{I}=\{I_1,\dots I_m\}} \sum_{j=1}^{m} \int_{I_j} (f(x) - \lambda) \ \mathrm{d}x$$

where the supremum is taken over all the families $\mathcal{I}$ of disjoint intervals $I_1, I_2 \dots I_m$. If X is a random variable with density f, it becomes

$$\mathbf{E}_f(\lambda) = \sup_{\mathcal{I}=\{I_1,\dots I_m\}} \sum_{j=1}^{m} (\mathbb{P}(X \in I_j) - \lambda |I_j|) = \sup_{\mathcal{I}=\{I_1,\dots I_m\}} \sum_{j=1}^{m} H_\lambda (I_j)$$

where $|I|$ is the length of the interval I (Lebesgue measure). Now, in practice, we have only n observations and the number of modes is unknown. So the authors introduce the following estimator where M is the assumed maximum number of modes:

$$\mathbf{E}_{n,M}(\lambda) = \sup_{\mathcal{I},\mathrm{Card}(\mathcal{I})\leq M} \sum_{j=1}^{\mathrm{Card}(\mathcal{I})} H_{n,\lambda}(I_j)$$

where $H_{n,\lambda}(I_j)$ approximates $H_\lambda(I_j)$ with an estimator of $\mathbb{P}(X \in I_j)$ based on the n observations (basically the histogram estimator).

Testing multimodality Müller and Sawitzki notice then that a density f has several modes when a large difference $\mathbf{D}_{n,M}(\lambda) = \mathbf{E}_{n,M}(\lambda) - \mathbf{E}_{n,1}(\lambda)$ occurs for some $\lambda > 0$. Indeed, when this gap is large, it means that $\mathbf{E}_{n,1}$ (the excess mass assuming a single mode) does not encompass enough probability at a given level λ, so another mode is

likely to exist. Thus the statistic $\Delta_{n,M} = \max_\lambda \mathbf{D}_{n,M}(\lambda)$ is suggested to build a test. The case $M = 2$ which refers to a "unimodality versus bimodality" test is particularly treated. The authors use critical values from the uniform distribution to finally build decision bound, assuming that the inequality

$$\mathbb{P}\left(\Delta_{n,2} \geq \kappa_\alpha \,|\, F\right) \leq \alpha$$

holds for all unimodal distributions F since n is large. Briefly, they estimate the distribution of $\Delta_{n,2}$ for the uniform distribution $U(0,1)$ through numerical simulations. An extension to test "bimodality vs higher order multimodality" is proposed based on a concentration bound for $\mathbb{P}\left(\Delta_{n,2} \geq \kappa\right)$.

Computing the excess mass All the given quantities are still theoretical, but methods to compute them are given in [92]. Especially, it exploits the following *splitting property*: to get the optimal intervals family $\mathcal{I}_{M+1}$ of $\mathbf{E}_{n,M+1}(\lambda)$ from those of $\mathbf{E}_{n,M}(\lambda)$, either a new interval I_{M+1} (disjoint from the others) is added or an interval $I_k \in \mathcal{I}_M$ is split (actually a sub-interval is removed from it). In the first case, we must look for the maximum of $H_{n,\lambda}(I)$ where I does not intersect the intervals of $\mathcal{I}_M$, so

$$A_M = \sup_{I, \forall j \leq M, I \cap I_j = \emptyset} H_{n,\lambda}(I).$$

In the second case, we must find the best interval I_k to split. For that purpose, as it boils down to remove a sub-interval I^c, we try to optimize the excess mass which is removed: $H_{n,\lambda}(I^c)$, so

$$B_M = \sup_{I^c \subset I_k, I_k \in \mathcal{I}_M} \left(-H_{n,\lambda}(I^c)\right).$$

Thus, the gap can be computed incrementally as follows

$$\mathbf{D}_{n,M}(\lambda) = \mathbf{E}_{n,M}(\lambda) - \mathbf{E}_{n,1}(\lambda) = \sum_{i=2}^{M} \left(\mathbf{E}_{n,i}(\lambda) - \mathbf{E}_{n,i-1}(\lambda)\right) = \sum_{i=2}^{M} \max\left\{A_i, B_i\right\}.$$

The optimization steps require to find intervals. Given some observations, these intervals have endpoints at data points, so these maximizations roughly need $O(n)$ computations.

5.3 The Dip test (1985)

The dip test is univariate unimodality test introduced by Hartigan and Hartigan [53]. Unlike the two other presented tests, the dip test of unimodality neither focuses on bumps, nor provides their localization by design (but some other contributions managed to extract such kind of information from it like [84]). The approach of Hartigan and Hartigan relies more on the definition of unimodality and is then more structural-based (in the algebraic point of view) while the ideas behind the Silverman's test and the excess mass test are rather geometrical-based. An `R` implementation of the dip test is also available on the official repository[2].

[2] `https://cran.r-project.org/package=diptest`

Dip statistic First let us recall the definition of a unimodal distribution (definition 5.1) initially given by Khintchine [64] and which can be found in [24].

DEFINITION 5.1 *A real random variable X or its cumulative distribution function F is called unimodal about a mode m if F is convex on $(-\infty, m)$ and concave on $(m, +\infty)$.*

Let us note $\mathbb{U}$, the set of the unimodal distributions. The *dip functional* is then defined by

$$D(F) = \inf_{U \in \mathbb{U}} \|F - U\|_\infty . \tag{5.1}$$

where $\|F - U\|_\infty = \sup_{x \in \mathbb{R}} |F(x) - U(x)|$ denotes the classical uniform norm of the real function $F - U$. Thus, the dip functional corresponds to the uniform distance between F and the set $\mathbb{U}$. Given n iid observations $X_1, X_2, \dots X_n$ from F, the *dip statistic* is merely the dip functional applied to the empirical distribution F_n: $\text{dip}(X_1, X_2, \dots X_n) = D(F_n)$. Besides, it has been noticed that the dip statistic and the excess mass are equivalent [17].

Bounding functions To compute $D(F_n)$, Hartigan and Hartigan need to get the closest unimodal distribution to F_n. For that purpose, they compute (several times) two functions bounding F_n on a given interval J. The upper bound is the *lowest concave majorant* (LCM) of F_n on J while the lower bound is the *greatest convex minorant* (GCM) of F_n on J. They are defined as follows

$$\text{LCM}(F): \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \inf\limits_{G \text{ concave}, G \geq F} G(x) \end{array} \qquad \text{GCM}(F): \begin{array}{ccc} \mathbb{R} & \to & \mathbb{R} \\ x & \mapsto & \sup\limits_{G \text{ convex}, G \leq F} G(x) \end{array}$$

where $F \leq G$ means that $\forall x \in J, F(x) \leq G(x)$. One can show that $\text{LCM}(F_n)$ and $\text{GCM}(F_n)$ are piecewise linear functions with points of contact with F_n at some X_i. The algorithms 5 and 6 detail a way to compute them (on an interval). It notably requires to previously get the ordered statistics $X_{(1)}, X_{(2)}, \dots X_{(n)}$ which are the observations sorted in ascending order. These algorithms return the sorted list of indices $\{i_1, i_2, \dots i_k\}$ which gives the points of contact with F_n. For instance if LCM returns $\{1, 15, 21\}$, it means that the least concave majorant is in contact with F_n at points $X_{(1)}, X_{(15)}$ and $X_{(21)}$. From these points we can easily compute the whole LCM (or GCM) by linear interpolation.

Computing the dip The algorithm 7 describes the way to compute the dip. The goal is to approximate F_n by two unimodal cdf which coincide with the GCM on an interval $(-\infty, m]$ and with the LCM on interval $[m, +\infty)$. The distance between F_n and this unimodal function is then the dip. The idea is to first build the GCM and LCM of F_n (lines 5 and 6). Then we look for the largest gap between these two functions (lines 7 to 15) which corresponds to twice the dip. Finally, the LCM and GCM are refined in this maximum gap zone to find the point where the gap between them is the highest and to refine these bounds locally in this area.

Algorithm 5 Least Concave Majorant

procedure LCM($X_{(1)}, X_{(2)}, \dots X_{(n)}, l, u$)
  LCM $\leftarrow \{u\}$
  while $l < u$ **do**
    $u \leftarrow l + \text{argmin}_{j \in [\![l, u-1]\!]} \frac{u/n - j/n}{X_{(u)} - X_{(j)}}$
    LCM $\leftarrow \{u\} \cup$ LCM
  end while
  return LCM
end procedure

Algorithm 6 Greatest Convex Minorant

procedure GCM($X_{(1)}, X_{(2)}, \dots X_{(n)}, l, u$)
  GCM $\leftarrow \{l\}$
  while $l < u$ **do**
    $l \leftarrow \text{argmin}_{j \in [\![l+1, u]\!]} \frac{(j-1)/n - (l-1)/n}{X_{(j)} - X_{(l)}}$
    GCM $\leftarrow$ GCM $\cup \{l\}$
  end while
  return GCM
end procedure

The dip test of unimodality In [53], the uniform distribution is considered as the least favorable unimodal distribution (like in the two aforementioned tests). For an empirical cdf F_n, the dip test compares $\text{dip}(F_n)$ with the quantiles of $\text{dip}(U_n)$ where U_n is the ecdf of n iid uniform random variables (these quantiles can be pre-computed with Monte-Carlo simulations). The dip test reports a $p-$value p which is the probability to have an ecdf U_n with $\text{dip}(U_n) > \text{dip}(F_n)$. The common threshold $\alpha = 0.05$ is used to make the decision: if $p < \alpha$ the ecdf F_n is probably multimodal.

5.4 Towards a multivariate test?

The main drawback of the previously detailed tests is that they are restricted to real valued random variable (dimension 1). Moreover they need to estimate the distribution (density or cdf) and may require several passes over the data. Some multivariate extensions have been proposed like the RUNT test [52] in 1992, or the MAP test [106] in 1994 which use heavy linkage structures and are then closely related to clustering algorithms. In addition, these two tests do not precise their unimodality definition, while it is required in the multivariate case. Eventually, despite these attempts to generalize unimodality testing, the computational cost of these approaches prevent them from being efficient for large sample sizes n.

Next, we explain the MAP test which performs better than the RUNT test [106]. First we detail the required linkage structure and then we derive the statistic used to

Algorithm 7 Dip statistic

procedure DIP($X_1, X_2, \dots X_n$)
  $X_{(1)}, X_{(2)}, \dots X_{(n)} \leftarrow \text{SORT}(X_1, X_2, \dots X_n)$
  $a \leftarrow 1, b \leftarrow n, D \leftarrow 0$
  while TRUE **do**
    $g \leftarrow \text{GCM}(X_{(1)}, X_{(2)}, \dots X_{(n)}, a, b)$
    $l \leftarrow \text{LCM}(X_{(1)}, X_{(2)}, \dots X_{(n)}, a, b)$
    $d_g, i^* \leftarrow \max_i |(g_i - 1)/n - L(g_i)|$ ▷ L is the linear interpolation from l
    $d_l, j^* \leftarrow \max_j |G(l_j) - j/n|$ ▷ G is the linear interpolation from g
    if $d_g > d_l$ **then**
      $d \leftarrow d_g$
      $a' \leftarrow g_{i^*}, b' \leftarrow \min_j \{l_j \,|\, g_{i^*} \leq l_j\}$
    else
      $d \leftarrow d_l$
      $a' \leftarrow \max_i \{g_i \,|\, g_i \leq l_{j^*}\}, b' \leftarrow l_{j^*}$
    end if
    if $d \leq D$ **then**
      return $D/2$
    else
      $D \leftarrow \max\left\{D, \sup_{x \in [X_{(a)}, X_{(a')}]} |G(x) - F(x)|, \sup_{x \in [X_{(b')}, X_{(b)}]} |L(x) - F(x)|\right\}$
      $a \leftarrow a', b \leftarrow b'$
    end if
  end while
end procedure

perform the test.

MAPST In [106], a spanning tree with a specific constraint is introduced: the Minimal Ascending Path Spanning Tree (MAPST) which is a spanning tree over the observations with a specific constraint (definition 5.2).

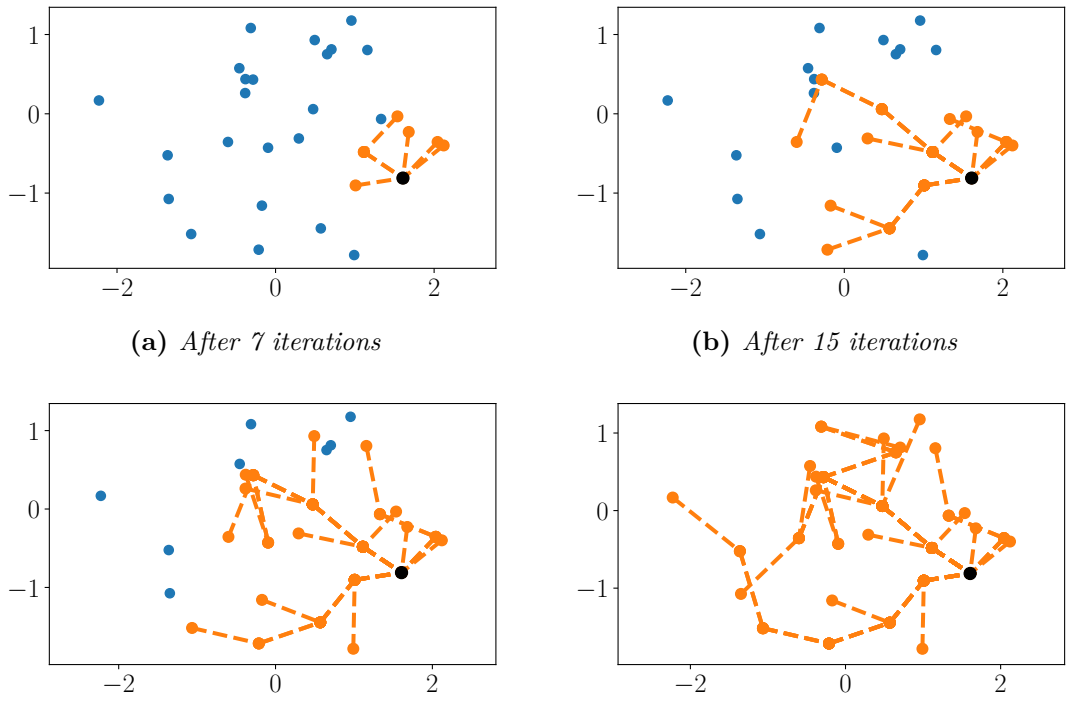
DEFINITION 5.2 *A MAPST with root node x_0 is a spanning tree of minimal total length constrained so that the link lengths are non-increasing in the path from any link to x_0.*

To build such a tree, Rozàl and Hartigan modify the Prim's algorithm [18], to take into account the ascending path (AP) constraint (algorithm 8)

A run of the MAPST algorithm is given on figure 5.3. At each iteration, a point is added so as to respect the two conditions: minimal length and path ascending property. Its computational complexity is $O(n \log n)$. In [106], the authors show that the MAPST algorithm constructs indeed a minimal ascending path spanning tree. Extensions of the MAPST algorithm, called *First* and *Second* algorithms, are also given to build multiple MAPSTs (MAPST with multiple root nodes). The idea is similar except that

Algorithm 8 MAPST

Require: $X_1, X_2, \dots X_n$
Ensure: MAPST of $X_1, X_2, \dots X_n$
1: Choose any point as a root node. Call this initial tree T_1.
2: At the $k-$th step, connect to T_{k-1} a point $X_i \notin T_{k-1}$ that will be connected with the shortest possible link provided that this new link is at least as long as the link to which it will be connected (Ascending Path Property).
3: Repeat Step 2 until all points have been included in the tree.



(a) *After 7 iterations*

(b) *After 15 iterations*

(c) *After 22 iterations*

(d) *After 30 iterations*

Figure 5.3: *MAPST algorithm on 30 points (the black point is the root)*

several points are taken at the beginning (roots). The algorithm will then build several disconnected sub-trees which are linked (bridge) in the end under the minimal length and AP constraints.

MAP statistic Once such structure is built, the authors define two measures

$$L_U(X_i) = \sum_{i \neq j} \log \rho_i(X_j) \qquad L_B(X_i) = \sum_{i_1, i_2 \neq j} \log \min_{i \in \{i_1, i_2\}} \rho_i(X_j) + \log d_{i_1, i_2}$$

where $\rho_i(x)$ is the length of the link between x and its successor in the direction of the root (denoted by X_i), and $d_{i,j}$ is the distance between the two subtrees (with root X_i and X_j respectively), that is to say the length of the shortest bridge between the two subtrees. The MAP statistic is then defined to test unimodality versus bimodality as follows:

$$\text{MAP} = \min_i L_U(X_i) - \min_{j,k} L_B(X_j, X_k)$$

In this context, computing the MAP requires to construct n single-root MAPSTs and $n(n-1)/2$ two-roots MAPSTs so the overall complexity is $O(n^3 \log n)$, therefore the test is not scalable. It can be extended to test multimodality by considering MAPSTs with more than two root nodes (the complexity becomes then $O(\binom{n}{k} n \log n)$ if k is the number of roots). The power of the test is experimented under two null hypothesis: the normal distribution $\mathcal{N}(0, I_d)$ and the uniform distribution on the unit sphere $\mathcal{U}(S_d)$.

5.5 Usage and related fields

Testing unimodality is generally not an end in itself. On the contrary, it may be used as a tool for related purposes. Particularly, some clustering algorithms use such approaches to find relevant groups of data. In [84], the dip statistics is used to find clusters in noisy data (*skinny-dip* algorithm). The authors idea is to find all the bumps in the ecdf because they represent "high" density regions. This information comes precisely from the dip computation. As the dip cannot be computed in the multivariate case, the authors apply it on each dimension. However it may create an issue, since a multimodal density can have unimodal margins (see figure 5.4). On the bottom left side, one can see that the bidimensional data is bimodal. However, its projection of the X (top) and Y (right) axis are perfectly unimodal, preventing to detect the multimodality of that dataset with unidimensional unimodality tests.

In [62], Kalogeratos and Likas present a combination of the dip statistics with the $k-$means algorithm (*dip-means* algorithm). More precisely, the dip test is performed on the pairwise distances of the points within a cluster. Indeed, a cluster is acceptable if it is unimodal (*unimodality assumption of the cluster*). If $k-$means provides a cluster which is not unimodal, dip-means re-run the algorithm assuming an additional cluster (with an initial state carefully chosen).

Other approaches to estimate the k of $k-$means exist but they are likely to run the clustering algorithm several times with different numbers of clusters so as to optimize a predefined criterion [59]. Examples of criteria are the Bayes Information Criterion (BIC), the Akaike Information Criterion (AIC), the Minimum Message Length (MML) [34], the Minimum Description Length (MDL) [51] or the Gap Statistics [124]. Some of them will be developed in the next part.

These tasks seem related to testing unimodality but they are slightly different. First, the objective of clustering is looking for the groups of data while unimodality testing decides whether data are gathered within a single group or not. However clustering needs more information: the number of clusters and/or some parameters to define the

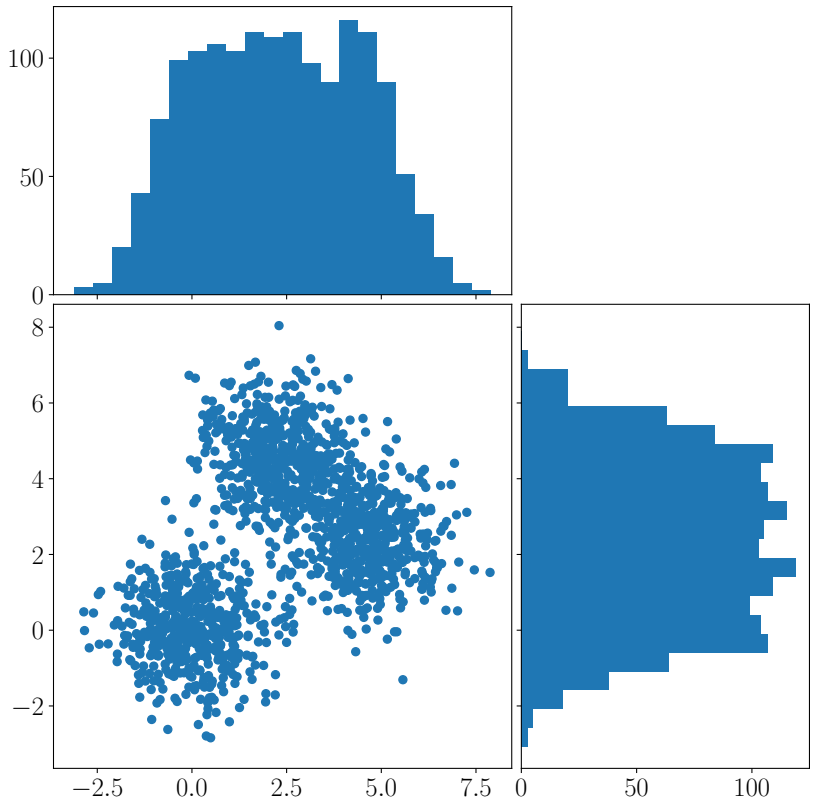


Figure 5.4: *Unidimensional test fails in 2D*

similarity between the observations. Thus, unimodality testing may be viewed as a simple macroscopic description of the data while clustering inspects it deeper.

5.6 Conclusion

In this chapter, the literature about unimodality testing has been detailed. We have noticed that the main tests only deal with univariate distributions which hinders their usage and the attempts to develop multivariate tests have been hampered by adding expensive structures.

Furthermore, these tests generally estimate the cdf (or the density). Unfortunately, this task requires either additional assumption (Silverman's test) or expensive computation (Excess Mass and Dip tests). In the latter case, the approaches cannot be easily adapted to streaming data. All these methods have also more technical drawbacks (especially about the significance) that we have not developed deeper as we prefer highlight structural aspects.

In the next chapter, we will develop the original idea behind our new test (the Folding Test of Unimodality) which aims to solve the aforementioned issues.

6

The Folding Mechanism

In this chapter, we present the core concept behind our new unimodality test: the *folding mechanism*. We especially introduce the *folding ratio*, the fundamental statistic of our developments, which relies on a simple geometrical transformation of the distribution. We will show how to easily compute it in both batch and streaming contexts. Furthermore, the folding mechanism will be developed on a theoretical case study and the powerful properties of the folding statistic will be presented.

Notations In the next parts we will use the following notations: $\| \cdot \|$ denotes the euclidean norm. In the general case, X is a random column vector of $\mathbb{R}^d$ ($d \in \mathbb{N}^*$), we note $\mathbb{E}(X) \in \mathbb{R}^d$ its expected value. We assume that X is $3-$integrable: it means that for all its components X_i, $\mathbb{E}(X_i^3)$ exists. We write $\Sigma(X) \in \mathbb{R}^{d \times d}$ (or merely Σ) its covariance matrix (we may use $\operatorname{Var} X$ in dimension 1). We assume that Σ is non-degenerated (so it is positive-definite and then invertible). If $A \in \mathbb{R}^a$ and $B \in \mathbb{R}^b$ are two random vectors, $\operatorname{Cov}(A, B) \in \mathbb{R}^{a \times b} = \mathbb{E}[(A - \mathbb{E}[A])(B - \mathbb{E}[B])^T]$ denotes their covariance. When X is a real valued random variable, we use also the common centered moments $M_k(X) = \mathbb{E}[(X - \mathbb{E}(X))^k]$. The trace of a square matrix $P \in \mathbb{R}^{n \times n}$, noted $\operatorname{Tr} P$, is equal to the sum of its diagonal elements.

The proposed algorithms may use the theoretical distribution of a random variable X, so we may use $\mathbb{E}(X), \operatorname{Var} X$ etc. but obviously they are valid on a given sample of n observations (drawn independently from the distribution of X). In this case, the common estimators have to be used:

$$\mathbb{E}(Z) \simeq \mu_Z = \frac{1}{n}\sum_{i=1}^{n} Z_i \quad \operatorname{Var}(Z) \simeq \frac{1}{n}\sum_{i=1}^{n}(Z_i - \mu_Z)^2 \quad \text{(dim. 1)}$$

$$\operatorname{Cov}(A, B) \simeq \frac{1}{n}\sum_{i=1}^{n}(A_i - \mu_A)\cdot(B_i - \mu_B)^T \quad (\text{so } \Sigma(Z) = \operatorname{Cov}(Z, Z))\,.$$

Thus, if $f(X)$ is a function applied on the theoretical random variable X, $f(X_1 \ldots, X_n)$ denotes its sample version (computed through the estimators given above).

6.1 General intuition

In this section, we present the general intuition of our approach aimed at testing whether a distribution is unimodal or not.

Univariate case Here we restrict the description of our approach to univariate distributions. The generalization will be made in the next paragraph.

Let us have a look at a bimodal density (figure 6.1a). In this case, the variance is likely to be "high" because the two modes make the data far from the expected value. Our idea is the following: if we fold up a mode to the other (with respect to the right **pivot** s^*), the resulting density (figure 6.1b) will have a far lower variance. Intuitively, this phenomenon will not appear for unimodal distributions (actually not with the same amplitude).

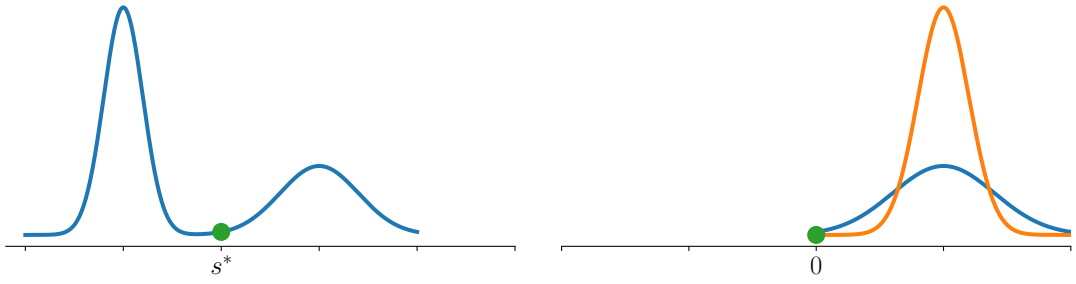
Let us sum up the approach: (1) find the right pivot s^*, (2) fold up the distribution along s^*, (3) compute the variance of the *folded distribution* and (4) compare it with the initial variance.

More formally, if we assume we get the pivot s^*, the folding step is performed with the transformation $X \mapsto |X - s^*|$, finally we will compute the **folding ratio**:

$$\varphi(X) = \frac{\operatorname{Var}|X - s^*|}{\operatorname{Var} X}. \quad \begin{array}{l} \swarrow \textit{folded variance} \\ \\ \nwarrow \textit{initial variance} \end{array}$$

Higher dimensions How can this approach be generalized? Actually, the transformation made on the distribution (the folding) is very similar except that the absolute value are replaced by the euclidean norm. Then we will consider $\operatorname{Var}\|X - s^*\|$ where X is a random vector of $\mathbb{R}^d$ and $s^* \in \mathbb{R}^d$ is the pivot.

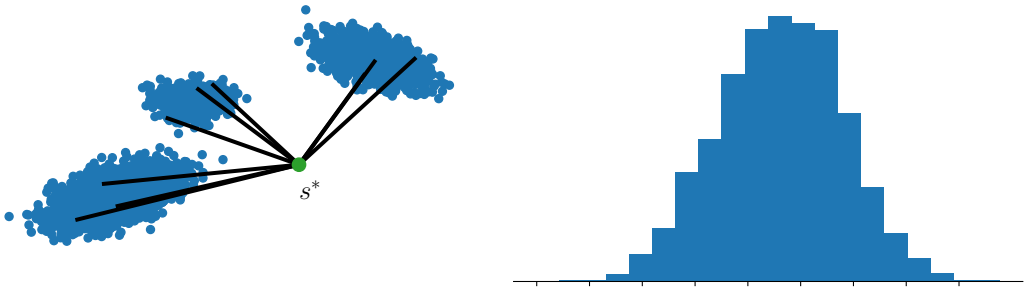
The figure 6.2 gives an empirical example of the folding mechanism in two dimensions. Given a trimodal distribution (figure 6.2a), and the right s^*, the folding $X \mapsto \|X - s^*\|$



(a) *Initial distribution* **(b)** *Folded distribution*

Figure 6.1: *Folding mechanism for univariate distribution*

turns the multi-modal bivariate distribution into a univariate distribution (figure 6.2b) which is likely to have a "low" variance. But obviously we have to mention about which reference this variance is low.



(a) *Initial distribution* **(b)** *Folded distribution*

Figure 6.2: *Folding mechanism in dimension 2*

The variance, which is (in dimension 1) the squared deviation of a random variable from its mean, is replaced by $\mathbb{E}\left[\|X - \mathbb{E}(X)\|^2\right]$ in higher dimensions (it corresponds to the trace of the covariance matrix). Indeed, in the unimodal case this expected value will be much lower than in the multimodal case. Thus, the folding step will potentially have more impact in the latter. Finally, the folding ratio is generalized through:

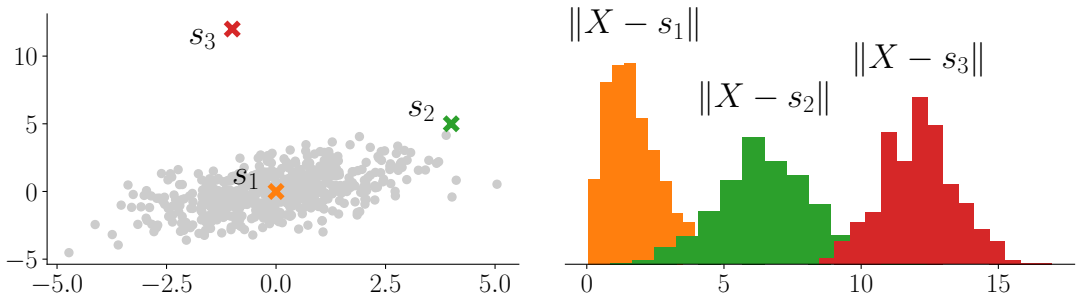
$$\varphi(X) = \frac{\operatorname{Var}\|X - s^*\|}{\mathbb{E}\left[\|X - \mathbb{E}(X)\|^2\right]} = \frac{\operatorname{Var}\|X - s^*\|}{\operatorname{Tr}\Sigma}.$$

The pivot s^* Until this part, we have assumed to get the *right* pivot s^*, i.e allowing the folding mechanism to significantly reduce the variance. Here we give more details about this pivot. Our goal is to find whether the variance may be significantly reduced

by folding. So, the natural way is to find the pivot which cuts down the variance the most, which is (when it exists):

$$s^* = \underset{s \in \mathbb{R}^d}{\operatorname{argmin}} \operatorname{Var} \|X - s\|.$$

This pivot is well-defined in dimension 1 but not in the general case (the minimum is not necessarily reached in higher dimensions) but we will try to get around this problem. A unimodal example in $\mathbb{R}^2$ is given below (figure 6.3).



(a) *Initial distribution* **(b)** *Folded distribution*

Figure 6.3: *Impact of the pivot location*

In the figure 6.3a, we draw a sample from a multivariate normal distribution (which is unimodal). Moreover, we choose three different pivots s_1, s_2 and s_3 and we plot the histogram of the folded observations $\|X - s_1\|, \|X - s_2\|$ and $\|X - s_3\|$. In this case, we may notice that s_1 seems to reduce the variance the best (so it may be the best pivot s^*). Actually, in the unimodal case, the best pivot s^* is likely to be close to the mode although in the multimodal case, it is likely to stand "between" the modes.

6.2 Formal approach

The previous paragraph introduced our main idea. Here, we develop it more formally. We introduce the function v_X defined by

$$v_X : \begin{array}{ccc} \mathbb{R}^d & \to & \mathbb{R} \\ s & \mapsto & \operatorname{Var} \|X - s\|. \end{array}$$

By definition, v_X is lower-bounded by 0, so we can define the folding ratio as below

$$\varphi(X) = \inf_{s \in \mathbb{R}^d} \frac{\operatorname{Var} \|X - s\|}{\mathbb{E}\left(\|X - \mathbb{E}(X)\|^2\right)}. \tag{6.1}$$

Thus, the computation of $\varphi(X)$ requires a minimization step, which is expensive. The ideal would be to have an expression of $\operatorname{argmin}_{s \in \mathbb{R}^d} v_X(s)$. Unfortunately, its existence

is not clear because v_X can be minimum to the infinity (we have noticed this divergence on some concrete and non absurd configurations).

In our approach, we aim to find a pivot even if the real one is not defined. Obviously this approximate pivot should produce a folding ratio which has intuitively the same properties as the theoretical one $\varphi(X)$. To circumvent the existence problem we rather use the following function:

$$v_X^{(2)} : \begin{array}{ccc} \mathbb{R}^d & \to & \mathbb{R} \\ s & \mapsto & \operatorname{Var}\left(\|X - s\|^2\right). \end{array}$$

If we find the s which minimizes $v_X^{(2)}$, we are likely to have a good candidate to minimize v_X. Moreover, the properties of $v_X^{(2)}$ are richer than the properties of v_X.

LEMMA 6.1 *For all $s \in \mathbb{R}^d$ we have:*

$$v_X^{(2)}(s) = 4\ s^T\, \Sigma(X) s - 4\ s^T \operatorname{Cov}\left(X, \|X\|^2\right) + \operatorname{Var}\left(\|X\|^2\right).$$

Proof. We have to notice that $\|X - s\|^2 = (X - s)^T (X - s) = \|X\|^2 - 2s^T X + \|s\|^2$. So

$$\begin{aligned} \operatorname{Var}\|X - s\|^2 &= \operatorname{Var}\left(\|X\|^2 - 2s^T X\right) \\ &= \operatorname{Var}\left(\|X\|^2\right) + 4 \operatorname{Var}\left(s^T X\right) - 4 \operatorname{Cov}\left(s^T X, \|X\|^2\right) \\ &= \operatorname{Var}\left(\|X\|^2\right) + 4\ s^T\, \Sigma(X) s - 4\ s^T \operatorname{Cov}\left(X, \|X\|^2\right). \end{aligned}$$

□

Lemma 6.1 tells us that $v_X^{(2)}$ is a quadratic form of $\mathbb{R}^d$. Thus, thanks to the properties of the covariance matrix Σ (positive-definite), $v_X^{(2)}$ is strictly convex and then it has a unique minimum, noted $s^*(X)$. Theorem 6.1 gives its analytical expression. In the following parts, we may use s^* (instead of $s^*(X)$) to ease the notation.

THEOREM 6.1 *The function $v_X^{(2)}$ has a unique minimum $s^*(X)$ given by:*

$$s^*(X) = \frac{1}{2}\, \Sigma^{-1}(X) \operatorname{Cov}\left(X, \|X\|^2\right). \tag{6.2}$$

In particular, if X is a real random variable (dimension 1) then

$$s^*(X) = \frac{1}{2}\, \frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \mathbb{E}(X) + \frac{1}{2} \frac{M_3(X)}{M_2(X)}. \tag{6.3}$$

Proof. We know that $v_X^{(2)}$ has a unique minimum. We just have to calculate the gradient of $v_X^{(2)}$ and find where it is zero. First of all,

$$\nabla_s v_X^{(2)}(s) = 8\, \Sigma(X) s - 4 \operatorname{Cov}\left(X, \|X\|^2\right).$$

As $\Sigma(X)$ is invertible we have

$$\nabla_s v_X^{(2)}(s^*) = 0 \quad \Longleftrightarrow \quad 8\,\Sigma(X)s^* = 4\,\mathrm{Cov}\left(X, \|X\|^2\right)$$
$$s^* = \frac{1}{2}\,\Sigma^{-1}(X)\,\mathrm{Cov}\left(X, \|X\|^2\right).$$

In the scalar case ($d = 1$), we have

$$\begin{aligned} s^* &= \frac{1}{2}\frac{\mathrm{Cov}\left(X, X^2\right)}{\mathrm{Var}\,X} \\ &= \frac{1}{2}\frac{\mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2)}{\mathbb{E}(X^2) - \mathbb{E}(X)^2} \\ &= \frac{1}{2}\frac{2\,\mathbb{E}(X)\left(\mathbb{E}(X^2) - \mathbb{E}(X)^2\right) + \mathbb{E}(X^3] - 3\,\mathbb{E}(X)\,\mathbb{E}(X^2) + 2\,\mathbb{E}(X)^3}{\mathbb{E}(X^2) - \mathbb{E}(X)^2} \\ &= \mathbb{E}(X) + \frac{1}{2}\frac{\mathbb{E}\left((X - \mathbb{E}(X))^3\right)}{\mathrm{Var}\,X} = \mathbb{E}(X) + \frac{1}{2}\frac{M_3(X)}{M_2(X)}. \end{aligned}$$

□

With the previous result, we have always a pivot allowing to compute an approximate of the folding ratio

$$\phi(X) = \frac{\mathrm{Var}\,\|X - s^*(X)\|}{\mathbb{E}\left(\|X - \mathbb{E}(X)\|^2\right)}. \tag{6.4}$$

The previous expression leads to a straightforward algorithm to compute $\phi(X)$ (algorithm 9). For simplicity reason, we will use "folding ratio" in the next paragraphs to express its approximate version.

Algorithm 9 Batch computation of $\phi(X)$

Require: X
Ensure: $\phi(X)$
  $D \leftarrow \mathbb{E}\left[\|X - \mathbb{E}(X)\|^2\right]$ ▷ or $D \leftarrow \mathrm{Tr}(\Sigma)$
  $s^* \leftarrow \frac{1}{2}\Sigma^{-1} \times \mathrm{Cov}\left(X, \|X\|^2\right)$
  return $\mathrm{Var}\,\|X - s^*\|/D$

We give some details about complexity. Let us consider a dataset of n observations in $\mathbb{R}^d$. The covariance matrix Σ needs $O(n \cdot d^2)$ operations and additional $O(d^3)$ operations are required for its inversion. Computing the norm of the observations and the covariance $\mathrm{Cov}\left(X, \|X\|^2\right)$ costs $O(n \cdot d)$ operations. Finally, another $O(d^2)$ operations are required for s^* and $O(n \cdot d)$ for the variance $\mathrm{Var}\,\|X - s^*\|$. In a nutshell, the complexity of the batch computation is linear for the number of observation n : $O(\underbrace{d^3 + n \cdot d^2}_{s^*} + \underbrace{n \cdot d}_{\mathrm{Var}\,\|X - s^*\|})$.

6.3 Incremental computation

In the section 6.2 we proposed a batch (or *offline*) version for the computation of $\phi(X_1 \ldots, X_n)$. However, in the streaming context, we can accelerate it with an incremental computation of s^*.

As seen in the Section 6.2, we have an analytical expression for the pivot: $s^* = \frac{1}{2}\Sigma^{-1}\operatorname{Cov}(X, \|X\|^2)$. Furthermore this expression can easily be computed incrementally or even updated over a sliding window. This is a very important property if we work on streaming data. The algorithm 10 details how to perform an update.

As an input we have a new incoming observation X_{new} and the square of its norm R_{new}. The idea is to compute their respective basic moments (cumulative sums S_X and S_R) and their co-moment (line 4) to finally compute an estimate of $\operatorname{Cov}(X, \|X\|^2)$ (line 7). About the inverse of the covariance matrix, we use the Sherman-Morrison formula [113] to compute the inverse of the co-moment of X (line 5) and then Σ^{-1} (line 6). We recall:

$$\operatorname{SM}(A^{-1}, u, v) = \left(A + u \times v^T\right)^{-1} = A^{-1} - \frac{A^{-1} \times u \times v^T \times A^{-1}}{1 + v^T \times A^{-1} \times u}.$$

Algorithm 10 Incremental computation of s^*

Require: $X_{\text{new}}, R_{\text{new}} = \|X_{\text{new}}\|^2$
Ensure: s^*
1: $n \leftarrow n + 1$
2: $S_X \leftarrow S_X + X_{\text{new}}$
3: $S_R \leftarrow S_R + R_{\text{new}}$
4: $V_{X,R} \leftarrow V_{X,R} + X_{\text{new}} \times R_{\text{new}}$
5: $V_X^{\text{inv}} \leftarrow \operatorname{SM}(V_X^{\text{inv}}, X_{\text{new}}, X_{\text{new}})$
6: $\Sigma^{\text{inv}} \leftarrow n \times \operatorname{SM}(V_X^{\text{inv}}, -S_X, \frac{1}{n}S_X)$ ▷ Σ^{-1}
7: $C \leftarrow \frac{1}{n}V_{X,R} - \frac{1}{n}S_X \times \frac{1}{n}S_R$ ▷ $\operatorname{Cov}(X, \|X\|^2)$
8: $s^* \leftarrow \frac{1}{2} \times \Sigma^{\text{inv}} \times C$

Here we present an update, so we have to deal with the initialization. Basically, the variables $n, S_X, S_R, V_{X,R}$ and C are set to 0. Unfortunately, the initialization of V_X^{inv} and Σ^{inv} is not as simple because it requires several observations X_i (at the beginning V_X^{inv} and Σ^{inv} are singular matrices). In practice, we start from a small batch of data X_{init}. Therefore, we can compute $V_X \leftarrow X_{\text{init}}^T \times X_{\text{init}}$ and then $V_X^{\text{inv}} \leftarrow V_X^{-1}$ (so Σ^{inv}). Finally, this update can easily be extended to the sliding window case requiring to store the quantities X_i and R_i, but here, we prefer not to overload the reader with additional formulas.

Is it possible to go further by computing $\phi(X)$ incrementally too? Unfortunately, this is not as straightforward. There is no problem for the denominator $\mathbb{E}\left[\|X - \mathbb{E}(X)\|^2\right]$ because this is the trace of Σ. The main issue comes from $\operatorname{Var}\|X - s^*\|$: a variance

can easily be computed incrementally (like in the algorithm 10) but s^* changes at each iteration, so an update of all the distances $\|X_i - s^*\|$ is needed.

In the paragraph 6.2, we discussed about the complexity of the batch computation. In table 6.1, we compare its complexity with the incremental computation in 3 cases:

- single computation: only $\phi(X_1, \dots X_n)$
- cumulative: $\phi(X_1), \phi(X_1, X_2) \dots \ \phi(X_1, \dots X_n)$
- sliding window: $\phi(X_1, \dots X_{1+w}), \dots \ \phi(X_n, \dots X_{n+w})$

Computation(s)	Batch	Incremental
Single	$d^3 + n \cdot d^2 \ (+ \ n \cdot d)$	
Cumulative	$n \cdot d^3 + n^2 \cdot d^2$	$d^3 + n \cdot d^2 + n^2 \cdot d$
Sliding window	$n \cdot d^3 + n \cdot w \cdot d^2$	$d^3 + n \cdot d^2 + n \cdot w \cdot d$

Table 6.1: *Complexity of each method (in big O notation)*

Without delving into the details, we may notice two phenomenons. First, the incremental computation of s^* decreases the dependency on the dimension d (actually the update of the matrix costs $O(d^2)$ instead of $O(d^3)$ in the batch version). Second, as the computation of $\mathrm{Var}\,\|X - s^*\|$ cannot be done efficiently, the computation of the folding statistics cannot be cut down anymore.

6.4 Theoretical case study

In this part, we focus on the unidimensional case through a theoretical study. Aside from our analysis, the paragraphs contain several mathematical proofs which aim to show a way to "concretely" manipulate the folding quantities but which are not a prerequisite for the next sections. Precisely, we calculate and analyze the folding values (s^*, ϕ ...) of the Beta distributions family, which gathers both unimodal and bimodal distributions.

Let us consider that X follows a beta distribution with parameters α, β (noted $\mathrm{Beta}(\alpha, \beta)$) whose density is defined on $[0, 1]$ (or $(0, 1)$) by

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha, \beta)}$$

with the expression of the beta function $\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\,\Gamma(\beta)}{\Gamma(\alpha+\beta)}$.

PROPOSITION 6.1 *Let $\alpha > 0$ and $\beta > 0$ and $X \sim \mathrm{Beta}(\alpha, \beta)$. The folding pivot of the Beta distribution is given by:*

$$s^*(X) = \frac{\alpha + 1}{\alpha + \beta + 2}.$$

Proof. As it is a common distribution we know that for all $k \in \mathbb{N}^*$,

$$\mathbb{E}(X^k) = \prod_{i=0}^{k-1} \frac{\alpha + i}{\alpha + \beta + i}.$$

So it notably gives

$$\mathbb{E}(X) = \frac{\alpha}{\alpha + \beta} \quad \mathbb{E}(X^2) = \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)} \quad \mathbb{E}(X^3) = \frac{\alpha+2}{\alpha+\beta+2}\,\mathbb{E}(X^2).$$

We can compute

$$\begin{aligned}
\operatorname{Cov}(X, X^2) = \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2) &= \mathbb{E}(X^2)\left(\frac{\alpha+2}{\alpha+\beta+2} - \frac{\alpha}{\alpha+\beta}\right) \\
&= \mathbb{E}(X^2)\frac{2\beta}{(\alpha+\beta)(\alpha+\beta+2)} \\
&= \frac{2\alpha\beta(\alpha+1)}{(\alpha+\beta)^2(\alpha+\beta+1)(\alpha+\beta+2)}.
\end{aligned}$$

As $\operatorname{Var} X = \dfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$ we finally have:

$$s^* = \frac{1}{2}\frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \frac{\alpha+1}{\alpha+\beta+2}.$$

□

Unfortunately the direct calculation of the folding ratio is not easy because of the expression $|X - s^*|$ which is hard to manipulate. However we can limit the study to the symmetrical case $\alpha = \beta$. In this case the density is symmetrical about $\frac{1}{2}$ and has an interesting evolving shape with α (see figure 6.4). When $\alpha < 1$, the density has a U shape (two modes in 0 and 1). For example, when $\alpha = \frac{1}{2}$, we get the arcsine distribution. The case $\alpha = 1$ corresponds to the uniform density and for $\alpha > 1$ we have a unimodal density.

PROPOSITION 6.2 *Let $\alpha > 0$. We suppose that X_α follows the symmetrical Beta distribution* $\operatorname{Beta}(\alpha, \alpha)$. *Therefore its folding ratio is given by:*

$$\phi(X_\alpha) = 1 - \frac{2\alpha + 1}{\alpha^2\,\mathrm{B}\left(\frac{1}{2}, \alpha\right)^2}.$$

Proof. As $\alpha = \beta$ we have:

$$s^* = \frac{\alpha+1}{\alpha+\beta+2} = \frac{1}{2}.$$

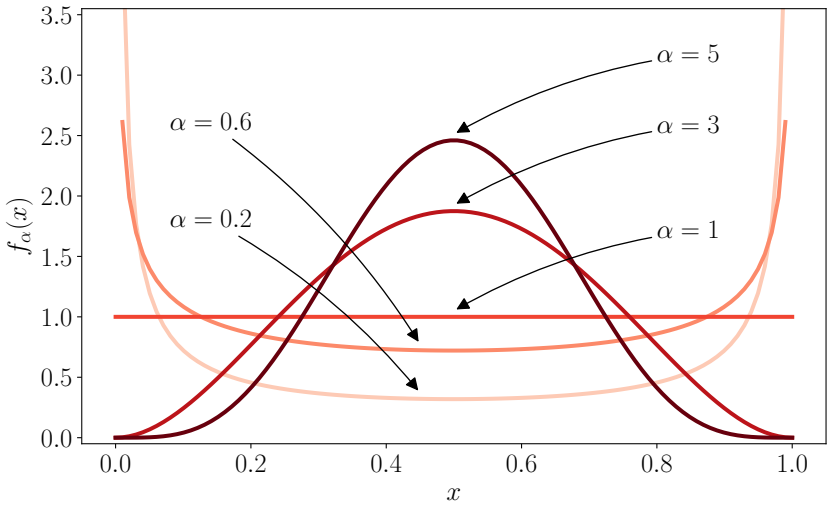


Figure 6.4: *Symmetrical beta density according to α*

Moreover we also have $\mathbb{E}[X_\alpha] = \frac{1}{2}$. Let us calculate the *folded* expected value:

$$\begin{aligned}
\mathbb{E}\left|X_\alpha - s^*\right| &= \frac{1}{\mathrm{B}(\alpha,\alpha)} \int_0^1 \left|x - \frac{1}{2}\right| (x(1-x))^{\alpha-1} \, \mathrm{d}x \\
&= \frac{2}{\mathrm{B}(\alpha,\alpha)} \int_0^{\frac{1}{2}} \left(\frac{1}{2} - x\right) (x(1-x))^{\alpha-1} \, \mathrm{d}x \quad \left(\text{sym. about } \frac{1}{2}\right) \\
&= \frac{1}{\mathrm{B}(\alpha,\alpha)} \int_0^{\frac{1}{2}} (1-2x) \, (x(1-x))^{\alpha-1} \, \mathrm{d}x.
\end{aligned}$$

A primitive of $x \mapsto (1-2x)\,(x(1-x))^{\alpha-1}$ is $x \mapsto \dfrac{1}{\alpha}\,(x(1-x))^{\alpha}$ so

$$\mathbb{E}\left|X_\alpha - s^*\right| = \frac{1}{\mathrm{B}(\alpha,\alpha)} \left[\frac{1}{\alpha}\,(x(1-x))^{\alpha}\right]_0^{\frac{1}{2}} = \frac{1}{\alpha 4^\alpha \,\mathrm{B}(\alpha,\alpha)}.$$

With the identity $\mathrm{B}\,(\alpha,\alpha) = \dfrac{2}{4^\alpha}\,\mathrm{B}\left(\dfrac{1}{2},\alpha\right)$, we finally get $\mathbb{E}\left|X_\alpha - s^*\right| = \dfrac{1}{2\alpha\,\mathrm{B}\left(\frac{1}{2},\alpha\right)}$.

We can notice that

$$\begin{aligned}
\mathrm{Var}\left|X_\alpha - s^*\right| &= \mathrm{Var}(X_\alpha - s^*) + \mathbb{E}(X_\alpha - s^*)^2 - \left(\mathbb{E}\left|X_\alpha - s^*\right|\right)^2 \\
&= \mathrm{Var}(X_\alpha) + \left(\mathbb{E}[X_\alpha] - \frac{1}{2}\right)^2 - \left(\mathbb{E}\left|X_\alpha - s^*\right|\right)^2 \\
&= \mathrm{Var}(X_\alpha) - \left(\mathbb{E}\left|X_\alpha - s^*\right|\right)^2.
\end{aligned}$$

Thus, the folding ratio is given by:

$$\phi(X_\alpha) = \frac{\operatorname{Var}|X_\alpha - s^*|}{\operatorname{Var} X_\alpha} = 1 - \frac{(\mathbb{E}\,|X_\alpha - s^*|)^2}{\operatorname{Var} X_\alpha} = 1 - \frac{2\alpha + 1}{\alpha^2\,\mathrm{B}\left(\frac{1}{2}, \alpha\right)^2}.$$

□

As the expression of $\phi(X_\alpha)$ is not so explicit, we plot its behavior according to α on figure 6.5. First, it is an increasing function therefore it confirms the main property of the folding ratio: the variance reduction is greater in the multimodal case. Moreover, it behaves as a continuous measure of unimodality: this phenomenon will be highlighted in the next paragraph.

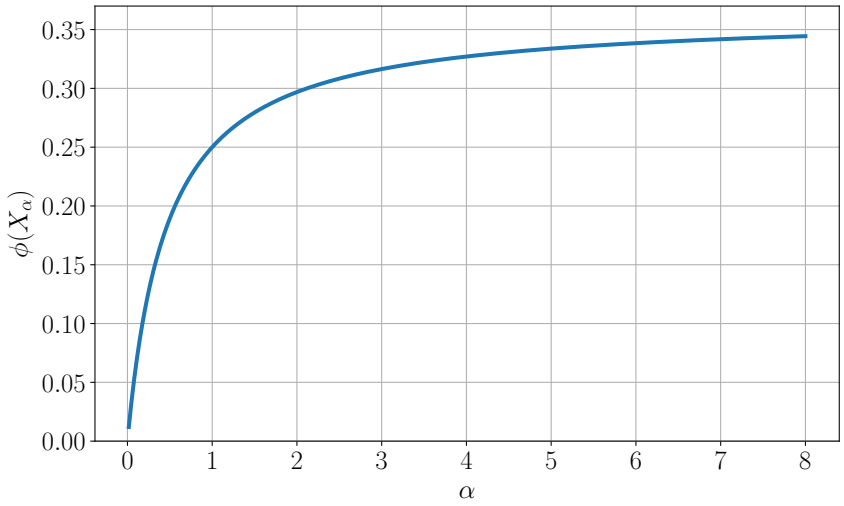


Figure 6.5: *The folding ratio of the symmetrical beta distribution according to α*

With the properties of the beta function, we can show mathematically that $\lim_{\alpha\to 0}\phi(X_\alpha) = 0$ and $\lim_{\alpha\to+\infty}\phi(X_\alpha) = 1 - \frac{2}{\pi}$. These results are quite interesting because they correspond respectively to the folding ratios of the Bernoulli and normal distributions (these values are given on table 6.2). Actually, the symmetrical beta density converges to the symmetrical Bernoulli distribution when $\alpha \to 0$ and has a shape close to the gaussian density when $\alpha \gg 1$. Furthermore, we can notice two things:

- $\phi(X_1) = \dfrac{1}{4}$ which is exactly the result of the uniform case;
- $\phi(X_{\frac{1}{2}}) = 1 - \dfrac{8}{\pi^2}$ which is exactly the result of the arcsine case.

6.5 Ranking the common distributions

In this paragraph, we study the folding values of some common univariate distributions. We show that the folding ratio $\phi(X)$ is relevant to *rank* the distributions according to their "unimodal character".

The expression of $s^*(X)$ is very convenient and allows us to calculate analytically its value (and also $\phi(X)$) for some well-known distributions. Particularly, several calculation examples can be found in appendix A.2. We have summarized them in table 6.2.

Distribution of X	$s^*(X)$	$\phi(X)$
Laplace $\mathcal{L}(\mu, b)$	μ	$\frac{1}{2}$
Exponential $\mathcal{E}(\lambda)$	$\frac{2}{\lambda}$	$1 - 4\mathrm{e}^{-2} - 4\mathrm{e}^{-4} \simeq 0.385$
Normal $\mathcal{N}(\mu, \sigma^2)$	μ	$1 - \frac{2}{\pi} \simeq 0.363$
Uniform $\mathcal{U}[a, b]$	$\frac{a+b}{2}$	$\frac{1}{4}$
Arcsine $\mathcal{A}[a, b]$	$\frac{a+b}{2}$	$1 - \frac{8}{\pi^2} \simeq 0.189$
Bernoulli $\mathcal{B}(p)$	$\frac{1}{2}$	0
Chi-squared $\chi^2(k)$	$k + 2$	$\simeq \frac{0.363k + 0.857}{k + 2.110}$
Gamma $\Gamma(k, \theta)$	$(k+1)\theta$	$\simeq \frac{0.363k + 0.427}{k + 1.05}$
Beta $\mathrm{Beta}(\alpha, \beta)$	$\frac{\alpha+1}{\alpha+\beta+2}$	$1 - \frac{2\alpha+1}{\alpha^2\,\mathrm{B}\left(\frac{1}{2}, \alpha\right)^2}$ (when $\alpha = \beta$)

Table 6.2: *Analytical folding ratio for some common distributions*

Distribution parameters impact First of all, we can notice that the folding ratios do not depend on the position and scale parameters of the distributions (identifiable through the common names $\mu, a, b, \lambda, \sigma$ in table 6.2). This property is very interesting because it

characterizes a whole distribution class through a single value (e.g. no matter μ, σ, the folding ratio of the normal distribution $\mathcal{N}(\mu, \sigma)$ remains $1 - \frac{2}{\pi}$).

Besides, we may notice that the folding ratio can depend on some other parameters, as in the gamma, beta or chi-squared cases. Actually, k (gamma, chi-squared) and α (beta) are both shape parameters. In that sense, it is intuitive to have this dependence on the folding ratio as they modify the shape of the densities.

To highlight our first two remarks, we may also check that the folding ratio of the gamma distribution depends on its shape parameter k but does not depend on its scale parameter θ.

We have observed that s^* and ϕ do not depend on position and scale parameters. Actually, we can mathematically prove it (see propositions 6.3 and 6.4).

PROPOSITION 6.3 *Let X be a $3-$integrable random vector of $\mathbb{R}^n$. For all $\lambda \in \mathbb{R}^*, b \in \mathbb{R}^n$, we have:*

$$s^*(\lambda X + b) = \lambda s^*(X) + b$$

Proof. First, we have $\Sigma(\lambda X + b) = \mathrm{Cov}(\lambda X + b, \lambda X + b) = \lambda^2 \Sigma(X)$. As $\lambda \neq 0$, we also get $\Sigma^{-1}(\lambda X + b) = \lambda^{-2} \Sigma^{-1}(X)$. Moreover,

$$\begin{aligned}
\mathrm{Cov}\left(\lambda X + b, \|\lambda X + b\|^2\right) &= \lambda \,\mathrm{Cov}\left(X, \lambda^2 \|X\|^2 + 2\lambda b^T X\right) \\
&= \lambda^3 \,\mathrm{Cov}\left(X, \|X\|^2\right) + 2\lambda^2 \,\mathrm{Cov}\left(X, b^T X\right) \\
&= \lambda^3 \,\mathrm{Cov}\left(X, \|X\|^2\right) + 2\lambda^2 \,\Sigma(X) b.
\end{aligned}$$

Finally,

$$\begin{aligned}
s^*(\lambda X + b) &= \frac{1}{2} \lambda^{-2} \Sigma^{-1}(X) \left(\lambda^3 \,\mathrm{Cov}\left(X, \|X\|^2\right) + 2\lambda^2 \,\Sigma(X) b\right) \\
&= \frac{1}{2} \lambda \,\Sigma(X) \,\mathrm{Cov}\left(X, \|X\|^2\right) + b \\
&= \lambda s^*(X) + b
\end{aligned}$$

□

PROPOSITION 6.4 *Let X be a $3-$integrable random vector of $\mathbb{R}^n$. For all $\lambda \in \mathbb{R}^*, b \in \mathbb{R}^n$, we have:*

$$\phi(\lambda X + b) = \phi(X)$$

Proof. We know that $\Sigma(\lambda X + b) = \lambda^2 \Sigma(X)$ so $\mathrm{Tr}\left(\Sigma(\lambda X + b)\right) = \lambda^2 \,\mathrm{Tr}\,\Sigma$. With the previous result we have $s^*(\lambda X + b) = \lambda s^*(X) + b$ so

$$\begin{aligned}
\phi(\lambda X + b) &= \frac{\mathrm{Var}\left|\lambda X + b - \lambda s^*(X) - b\right|}{\lambda^2 \,\mathrm{Tr}\,\Sigma} \\
&= \frac{\lambda^2 \,\mathrm{Var}\left|X - s^*(X)\right|}{\lambda^2 \,\mathrm{Tr}\,\Sigma} = \phi(X)
\end{aligned}$$

□

Links between distributions. The continuity of the folding ratio allows the transport of important properties. The chi-squared and the exponential distributions are both a special case of the gamma distribution. By setting $k = 1$ and $\theta = \frac{1}{\lambda}$ on the gamma distribution, we get the exponential $\mathcal{E}(\lambda)$ and, by setting $\theta = 2$ and applying the transformation $k \mapsto \frac{k}{2}$ on the gamma parameters, we get the chi-squared $\chi^2(k)$. The reader could then check that these dependencies are found on the folding values. This remark has already been made in the study of the beta distribution (the arcsine distribution is one of its special case).

Let us assume that $X_{k,\theta} \sim \Gamma(k, \theta)$. If we look more carefully on the approximate expression of folding ratio of $X_{k,\theta}$, we can notice that (independently from θ) $\lim_{k\to\infty} \phi(X_{k,\theta}) = 0.363$ which seems very close to the folding ratio of the normal distribution. Here, the approximation of $\phi(X_{k,\theta})$ prevents us from proving formally the equality, however it is true as for large k the gamma distribution converges to normal distribution.

A ranking of the common univariate distributions The figure 6.6 ranks the distribution according to their folding ratios while the figure 6.7 shows some plots of the corresponding densities. On the figure 6.6 we notice that the presented distributions have fixed folding ratio values except two of them (beta and gamma). As we have seen previously, their folding ratio lie in some ranges (visible in red and blue respectively) and change according to their shape parameters.

If we merely compare the values, the distributions seem to be well-ranked according to their "unimodal character". Obviously, we concede this is not a well-defined notion (more a visual aspect) but we can sketch what we mean: a distribution is more unimodal when the density at its folding pivot is sharper.

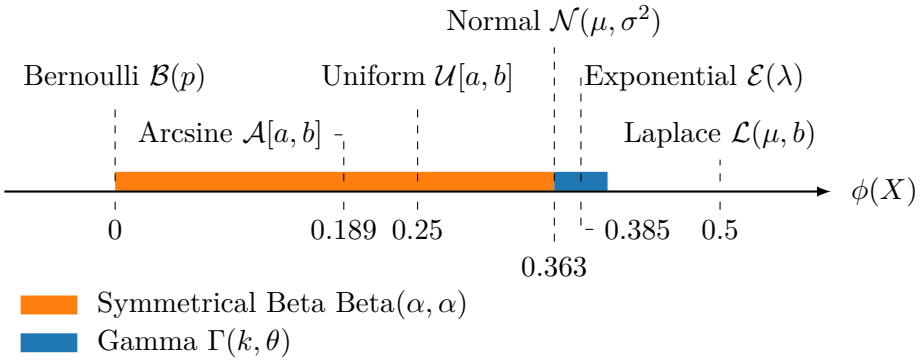


Figure 6.6: *Ranking of some common univariate distributions according to their folding ratio*

Table 6.2 presents only common univariate distributions. Actually, analytical expressions for more complex distributions (e.g. multivariate) are more difficult to get.

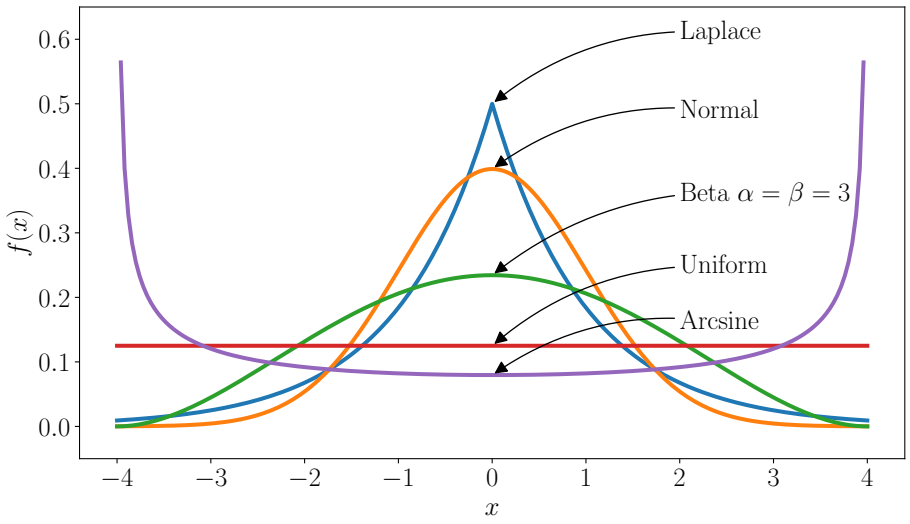


Figure 6.7: *Plot of some univariate densities: the sharper is the mode, the greater is the folding ratio*

Concluding remarks In this chapter, we have presented the folding mechanism. In particular we have highlighted the properties of the folding pivot s^* and the folding ratio ϕ.

The approximations used for s^* make it incrementally computable leading to a fast two-passes algorithm to compute ϕ. Moreover we have seen that this folding ratio does not depend on the location and scale parameters of the distribution and has a powerful distribution ranking capacity. In the next chapter, this material will be used so as to build the Folding Test of Unimodality (FTU).

7

The Folding Test of Unimodality

In the previous chapter, we have developed the folding ratio: a relevant statistic to rank the distributions according to their unimodality character. Henceforth, we want to make a decision: is the distribution of X unimodal or not? Here, we describe the folding test of unimodality (FTU) which is based on the folding mechanism and we investigate its statistical significance. Precisely, we detail our approach aimed to efficiently compute decision bounds and $p-$values. Finally, we show the limits of the test on symmetrical configurations.

7.1 Definition of the folding test of unimodality

We have seen the unimodality ranking power of the folding statistic. To answer, we need to compare the folding ratio $\phi(X)$ to a reference. As claimed by Hartigan's work [53] and also observed in the paragraph above, the uniform distribution is commonly considered as this reference.

Actually, there is a mathematical argument to consider it as a limit case. Let us note $\mathbb{U}$ the set of the univariate unimodal distributions (with mode at 0) and, for all $a \in \mathbb{R}$, W_a the uniform distribution on $[0, a]$ (or $[a, 0]$ if $a < 0$). It is shown that $\mathbb{U}$ is a convex

set and that the $W_a, a \in \mathbb{R}$ are in the boundaries of this set. More than that, we have the equality $\mathbb{U} = \text{ConvexHull}\{W_a, a \in \mathbb{R}\}$ meaning that these boundaries are generated by the uniform distributions. The reader could have a look at [24] for further details.

In our work, we generalize this approach for all dimensions $d \in \mathbb{N}^*$. We consider the uniform distribution within the $d-$ball as the limit case of unimodality. We do not need to precise the radius of this $d-$ball because the folding ratio does not depend on it (see proposition 7.1).

PROPOSITION 7.1 *Let $R > 0$ and $B_d(R) = \{x \in \mathbb{R}^d \mid \|x\| \leq R\}$ the $d-$dimensional ball of radius R. The approximate folding ratio of the uniform random vector $U_d \sim \mathcal{U}(B_d(R))$ is:*

$$\phi_d = \phi(U_d) = \frac{1}{(d+1)^2}.$$

Proof. Let $U_d \sim \mathcal{U}(B_d(R))$. For symmetry reasons, we naturally have $s^* = 0$. Now, if we note F_d the cumulative distribution of $\|U_d\|$, its expression is given by the ratio of the hyperspheres of radius r and radius R, so

$$\forall r \in [0, R], \quad F_d(r) = \mathbb{P}(\|U_d\| < r) = \left(\frac{r}{R}\right)^d.$$

Then, the density of $\|U_d\|$ is

$$\forall r \in [0, R], \quad f_d(r) = \frac{d}{R}\left(\frac{r}{R}\right)^{d-1}.$$

Therefore, we can calculate the moments of $\|U_d\|$:

$$\mathbb{E}(\|U_d\|) = \int_0^R r f_d(r)\, \mathrm{d}r = d\int_0^R \left(\frac{r}{R}\right)^d \mathrm{d}r = Rd\int_0^1 u^d\, \mathrm{d}u = R\frac{d}{d+1}$$
$$\mathbb{E}\left(\|U_d\|^2\right) = \int_0^R r^2 f_d(r)\, \mathrm{d}r = Rd\int_0^R \left(\frac{r}{R}\right)^{d+1} \mathrm{d}r = R^2 d\int_0^1 u^{d+1}\, \mathrm{d}u = R^2\frac{d}{d+2}$$

It gives

$$\begin{aligned}\text{Var}\,\|U_d - s^*\| = \text{Var}\,\|U_d\| &= \mathbb{E}\left(\|U_d\|^2\right) - \mathbb{E}(\|U_d\|)^2 \\ &= R^2\frac{d}{d+2} - R^2\left(\frac{d}{d+1}\right)^2 \\ &= \frac{R^2 d}{(d+2)(d+1)^2}.\end{aligned}$$

Moreover $\mathbb{E}\left(\|U_d - \mathbb{E}(U_d)\|^2\right) = \mathbb{E}\left(\|U_d\|^2\right)$ so

$$\phi(U_d) = \frac{R^2 d}{(d+2)(d+1)^2}\frac{1}{R^2\frac{d}{d+2}} = \frac{1}{(d+1)^2}$$

□

The power of our method relies on its independence to distribution parameters, allowing to catch whole classes through a single value. As of now, we are able to build a unimodality test based on the comparison of folding ratios.

DEFINITION 7.1 *Let X be a 3-integrable random vector of $\mathbb{R}^d$. We define the* ***folding statistics*** *$\Phi(X)$ by*

$$\Phi(X) = \frac{\phi(X)}{\phi_d} = (1+d)^2\ \phi(X). \tag{7.1}$$

The latter definition leads then to the folding test of unimodality:

DEFINITION 7.2 (FTU: Folding Test of Unimodality) *Let X be a 3-integrable random vector of $\mathbb{R}^d$.*
If $\Phi(X) \geq 1$, the distribution of X is unimodal.
If $\Phi(X) < 1$, the distribution of X is multimodal.

7.2 Statistical significance

Obviously, we do not have the same level of confidence in the test whether we have 100 or 1 billion observations. Here we precise the decision bounds when the sample size is n.

Let us take a sample $X_1 \ldots, X_n \in \mathbb{R}^d$. We have seen how we can compute $\Phi(X_1 \ldots, X_n)$. According to this value we can decide whether the distribution is unimodal or multimodal. Now, let us imagine that the sample $U_1 \ldots, U_n$ is drawn from the uniform distribution: the test becomes very uncertain because the computed folding statistics would be close to 1. So we may understand that the test is more significant if we are far from the uniform case.

This classical statistical hypothesis testing problem leads us to provide $p-$values. For instance, let $q > 0$ and let us assume we have computed $\Phi(X_1 \ldots, X_n) = 1 - q < 0$, so the distribution is considered as multimodal. The probability to have a uniform example $U_1 \ldots, U_n$ with a lower folding statistics is given by:

$$\mathbb{P}\left(\Phi(U_1 \ldots, U_n) < 1 - q\right) = \mathbb{P}\left(1 - \Phi(U_1 \ldots, U_n) > q\right).$$

Conversely, we can imagine that we have computed $\Phi(X_1 \ldots, X_n) = 1 + q$. In this case, we can estimate the probability to have a uniform sample with a higher folding statistics:

$$\mathbb{P}\left(\Phi(U_1 \ldots, U_n) > 1 + q\right) = \mathbb{P}\left(\Phi(U_1 \ldots, U_n) - 1 > q\right).$$

Obviously we want these two probabilities to be low. It means that the $p-$value $p = \mathbb{P}\left(|\Phi(U_1 \ldots, U_n) - 1| > q\right)$ must be as low as possible. In a nutshell, if we want a significant test (usually $p \leq 0.05$), it implies to choose q high enough, making the uncertainty area $1 \pm q$ wider. Conversely, if the test outputs a value $\Phi(X_1 \ldots, X_n)$, it leads to a decision whose significance can be estimated by p (the lower it is, the more significant it will be).

If we note $Y_{d,n} = |\Phi(U_1 \dots, U_n) - 1|$, the ideal would be to know the distribution of $Y_{d,n}$. However $Y_{d,n}$ does not follow a common distribution, leading us to compute some quantiles with Monte-Carlo simulations. Our goal is to provide an interpolated formula $q(d, n, p)$ verifying $\mathbb{P}(Y_{d,n} > q(d, n, p)) = p$. Then it will give the right decision bound according to the context d, n and the desired $p-$value. Conversely, it is also relevant to have an "inverted" formula to compute the $p-$value according to the context and the output folding statistics.

Our complete approach is detailed in appendix A.3. Through all our simulation results (about 20000 (n, d, p) configurations) we get the following formula:

$$q(n, d, p) = \frac{a\,(p - b\log(1-p))\,(\log(d) + c)}{\sqrt{n}}, \tag{7.2}$$

with $a = 0.479, b = 0.407$ and $c = 2.029$.

Given a dataset X (n observations in dimension d) and $p-$value p, we can compute the decision boundary (the quantile q) which provides the following property: $\mathbb{P}\left(|\Phi(U_1 \dots, U_n) - 1| > q\right) = p$. Conversely, if we compute the folding statistics $\Phi = 1 \pm q$, we can also compute the significance of the test (the $p-$value) in solving

$$\begin{aligned} \frac{a\,(p - b\log(1-p))\,(\log(d) + c)}{\sqrt{n}} &= q = |\Phi - 1| \\ \iff \quad p - b\log(1-p) &= \frac{|\Phi - 1|\sqrt{n}}{\log(d) + c} \end{aligned} \tag{7.3}$$

As the function $p \mapsto p - b\log(1-p)$ is a bijection from $[0, 1[$ to $[0, +\infty[$, the above equation has a unique solution. We can find it numerically with a common root search algorithm. However, if we assume that p is small (if Φ is quite far from 1) we have $\log(1-p) \simeq -p$ so

$$p \simeq \frac{|\Phi - 1|\sqrt{n}}{(\log(d) + c)\,(1 + b)} \tag{7.4}$$

7.3 Limits

Previously we have given many clues showing that the folding ratio is relevant to quantify the unimodality of probability distributions. It has lead us to a test which merely outputs the position of the tested distribution about the uniform case (see figure 6.6). However our experiments have demonstrated that the test may fail in some configurations, considering a multimodal distribution as unimodal. Particularly, the test is not adapted when a multimodal distribution has some symmetries in it.

Let us treat a special distribution that we call *three point masses* and that we note $\mathrm{TPM}_{a,\varepsilon}$. This distribution has two parameters $a \geq 0$ and $|\varepsilon| \leq a$. The distribution is defined in table 7.1 and graphically represented on figure 7.1. This distributions is clearly multimodal once we assume $a > 0$. For symmetry reasons we restrict the distribution to positive values, but we can naturally extend our results when $-a \leq \varepsilon \leq 0$.

X	$-a$	ε	a
$\mathbb{P}(X)$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Table 7.1: *Definition of* $\mathrm{TPM}_{\mathrm{a},\varepsilon}$

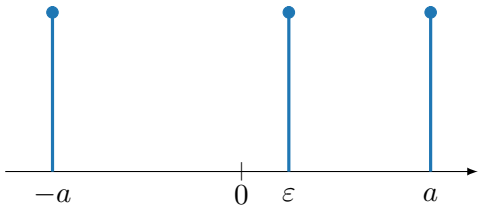


Figure 7.1: *Distribution* $\mathrm{TPM}_{\mathrm{a},\varepsilon}$

PROPOSITION 7.2 *Let* $a \geq \varepsilon \geq 0$ *and let* $X \sim \mathrm{TPM}_{\mathrm{a},\varepsilon}$. *The folding pivot of* $\mathrm{TPM}_{\mathrm{a},\varepsilon}$ *is*

$$s^*(X) = \frac{1}{2}\,\varepsilon\,\frac{\varepsilon^2 - a^2}{\varepsilon^2 + 3a^2} \tag{7.5}$$

and the folding ratio depends only on $r = \varepsilon/a$ *and is given by*

$$\phi(X) = \frac{(1-r)^2\left(9 - 3r + 4r^2 + r^3 + r^4\right)}{(3+r^2)^3}. \tag{7.6}$$

Proof. First we need to calculate the moments of X:

$$\begin{aligned}
\mathbb{E}(X) &= \frac{1}{3}\left(-a + \varepsilon + a\right) = \frac{1}{3}\,\varepsilon \\
\mathbb{E}(X^2) &= \frac{1}{3}\left(a^2 + \varepsilon^2 + a^2\right) = \frac{1}{3}\left(2a^2 + \varepsilon^2\right) \\
\mathbb{E}(X^3) &= \frac{1}{3}\left(-a^3 + \varepsilon^3 + a^3\right) = \frac{1}{3}\,\varepsilon^3.
\end{aligned}$$

So it leads to

$$\begin{aligned}
\operatorname{Var} X = \mathbb{E}(X^2) - \mathbb{E}(X)^2 &= \frac{1}{9}\left(6a^2 + 3\varepsilon^2 - \varepsilon^2\right) = \frac{2}{9}\left(3a^2 + \varepsilon^2\right) \\
\operatorname{Cov}(X, X^2) = \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2) &= \frac{1}{9}\left(3\varepsilon^3 - 2a^2\varepsilon - \varepsilon^3\right) = \frac{2}{9}\left(\varepsilon^3 - a^2\varepsilon\right).
\end{aligned}$$

Finally we get the folding pivot

$$s^* = \frac{1}{2}\frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \frac{1}{2}\,\varepsilon\,\frac{\varepsilon^2 - a^2}{3a^2 + \varepsilon^2}.$$

□

Here we do not detail how we can get the formulas (7.6) but we precise the information it contains. First we notice that the folding ratio only depends on the ratio ε/a. Once more, we experiment its powerful properties.

To understand the limits of the test we plot the folding statistics $\Phi(X_r)$ according to r on figure 7.2, where X_r is a $\mathrm{TPM}_{\mathrm{a},\varepsilon}$ random variable with $r = \varepsilon/a$. The folding statistics is computed from the folding ratio with the formula (7.1) as $d = 1$ here. The

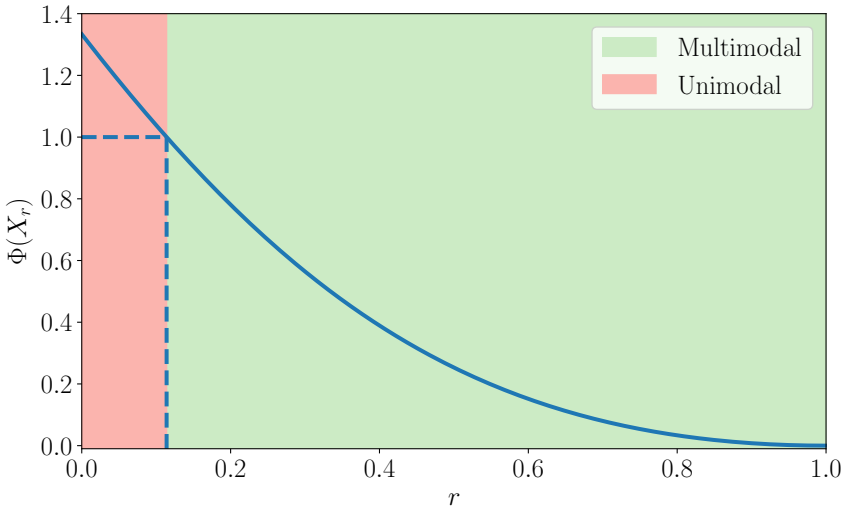


Figure 7.2: *Folding statistics* $\Phi(X_r)$ *according to* $r = \frac{\varepsilon}{a}$

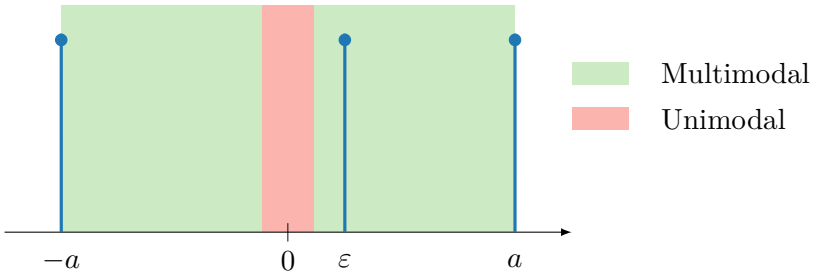


Figure 7.3: *Output of the FTU on* $\mathrm{TPM}_{\mathrm{a},\varepsilon}$ *according to* ε

real issue is the following: an interval $L = [0, r_{\lim}]$ exists such that the test ouputs that X_r is unimodal when $r \in L$. In particular, we have $r_{\lim} \simeq 0.114$.

Therefore if r is too close to zero, the test may fail. In fact, "zero" represents here the middle of the two extreme modes (at a and $-a$). It means that when the middle mode (at ε) is too close to the middle of the two other modes, the FTU may be wrong.

This distribution is clearly a special case but we extend it to a more general caveat: the folding test of unimodality can suffer from the symmetries of the modes.

However we also has given the folding ratio of the Bernoulli distribution (see table 6.2) which corresponds to a "two point masses". This distribution is obviously symmetrical and the test rightly outputs $\Phi(X) = 0$ (because $\phi(X) = 0$). If we investigate deeper, we may notice that the difference lies in the position of the folding pivot in relation to a mode. In the three point masses example, the test fails because the folding pivot is close to a mode, exactly as if we treat a unimodal density.

Now let us rewrite our failing condition $r < r_{\lim}$ to a condition on the relative distance

of s^* to the mode at ε, noted d_r.

$$d_r = \frac{|s^* - \varepsilon|}{2a} = \frac{1}{4}\, r\, \frac{r^2+7}{r^2+3}.$$

We can show that d_r is a strictly increasing function of $r \in [0,1]$. Thus,

$$0 \leq r \leq r_{\text{lim}} \iff 0 \leq d_r \leq \frac{1}{4}\, r_{\text{lim}}\, \frac{r_{\text{lim}}^2+7}{r_{\text{lim}}^2+3} \simeq 0.05.$$

This latter result quantifies how far from a mode the pivot should be. In a nutshell, the FTU can fail if the pivot s^* is at a distance from a mode lower than 5% of the size of the distribution support (here represented by the term $2a$).

The natural question to avoid this problem would be the following: can we artificially move the folding pivot s^* to avoid this critical area? In some cases it may solve the problem but unfortunately, in the general case, the variance reduction would also be impacted and the final decision would not be improved.

Even if this limiting example is a special case which does not seem relevant in real world context, we look for a more robust procedure, warning of symmetrical configurations.

7.4 Experiments

In this section, we highlight the relevance of the folding test of unimodality. First of all, we detail the implementations we provide. Then we experiment the FTU on real world multidimensional data to emphasize its correctness and its practical uses. Particularly, we show that it gives a paramount statistical description necessary for data analysis.

Implementations Several implementations are available to perform the folding test of unimodality: `C++`, `python3` and `R`. All the required information to use them are at `https://asiffer.github.io/libfolding/`. The initial work is the `C++` library called `libfolding` which implements the test for either batch or streaming data. The source code is downloadable at `https://github.com/asiffer/libfolding` but to ease its installation a `debian` package (with the same name) is also available (`ppa:asiffer/libfolding`). Upon this library we provide a `python3` wrapper (source code at `https://github.com/asiffer/python3-libfolding`). This library naturally includes the `python3` bindings to `libfolding` but also a pure `python3` implementation of the FTU for the batch version. This implementation is also available through a `debian` package called `python3-libfolding` (at the same ppa as `libfolding`). Finally we have developed a pure `R` package called `Rfolding` which implements the test only in the batch version. This package has been accepted to CRAN (`https://cran.r-project.org/package=Rfolding`), therefore the community can easily download and use it too. In the following experiments we mostly use the `C++` library.

Should I try to cluster Pokémon? In this section, we will show that our test is able to avoid a useless clustering step. Particularly, we analyze the Pokemon Stats Dataset from `kaggle`[1]. This dataset gathers statistics of the Pokémon until the 6^{th} generation: it includes 21 variables per each of the 721 Pokémons. In this short experiment, we keep only the 6 basic fight statistics: *Attack*, *Defense*, *HealthPoints*, *SpecialAttack*, *SpecialDefense* and *Speed*.

A priori, in such a 6-dimensional space, this is not easy to claim whether some clusters arise. Either all the features pairs could be plotted, or a clustering algorithm could be run. Through the first method (figure 7.4), the distributions of the features and their pairs lead us to think that the whole distribution is rather unimodal.

When we compute the folding statistics to this dataset, we get $\Phi(X) = 5.04$. As guessed above, this result claims that the distribution is unimodal.

To show that our approach is relevant, we compare the result of our test with the output of some well-known clustering algorithms. Unfortunately, most of them needs to set precisely the number of clusters, so we only test the approaches having some procedures to find it.

First we try to model the data with gaussian mixtures. The number of clusters is found by minimizing either the Akaike's or the Bayesian Information Criterion (AIC, BIC). Then we use the *gap statistics* [124] of Tibshirani *et al.*, which is a common statistics to estimate the number of clusters. We test also the Mean Shift algorithm [20] with different quantiles $q \in [0.3, 0.8]$. This parameter tunes the "similarity" of the points (it sets the bandwidth of the underlying gaussian kernel). Finally, we run the Affinity Propagation algorithm [37] with the euclidean distance as similarity. For these two last approaches, we show only "stable" results, so we do not present neither absurd parameter choices (the mean shift algorithm with $q \to 1$ would make all the points similar) nor absurd outputs of some algorithms (the affinity propagation algorithm with the gaussian similarity did not output less than 49 clusters). Finally we compute the average silhouette score [104] to estimate the clustering quality (the closer to 1, the better).

The results are given in the table 7.2. We can notice that there is no consensus among the different methods: they do not output the same number of clusters. At the very least, this indicates that the dataset does not contain obvious clusters. Now regarding the returned clusters, the average silhouette scores are mostly low, even close to zero, meaning that these clusters are not well separated. The highest silhouette score is reached by Mean Shift, however the algorithm returned 3 clusters of size 717-3-1, so in practice it found only one cluster. These results are in favor of the unimodality of the data, agreeing with the result of our folding test of unimodality.

Before running clustering algorithms, our simple test can thus be a crucial step to decide whether clustering is relevant or not.

Stock market behaviors The aim of this section is twofold. Firstly, we analyze the behavior of the folding statistics over a sliding window. Secondly, we show how the

[1] `https://www.kaggle.com/alopez247/pokemon`

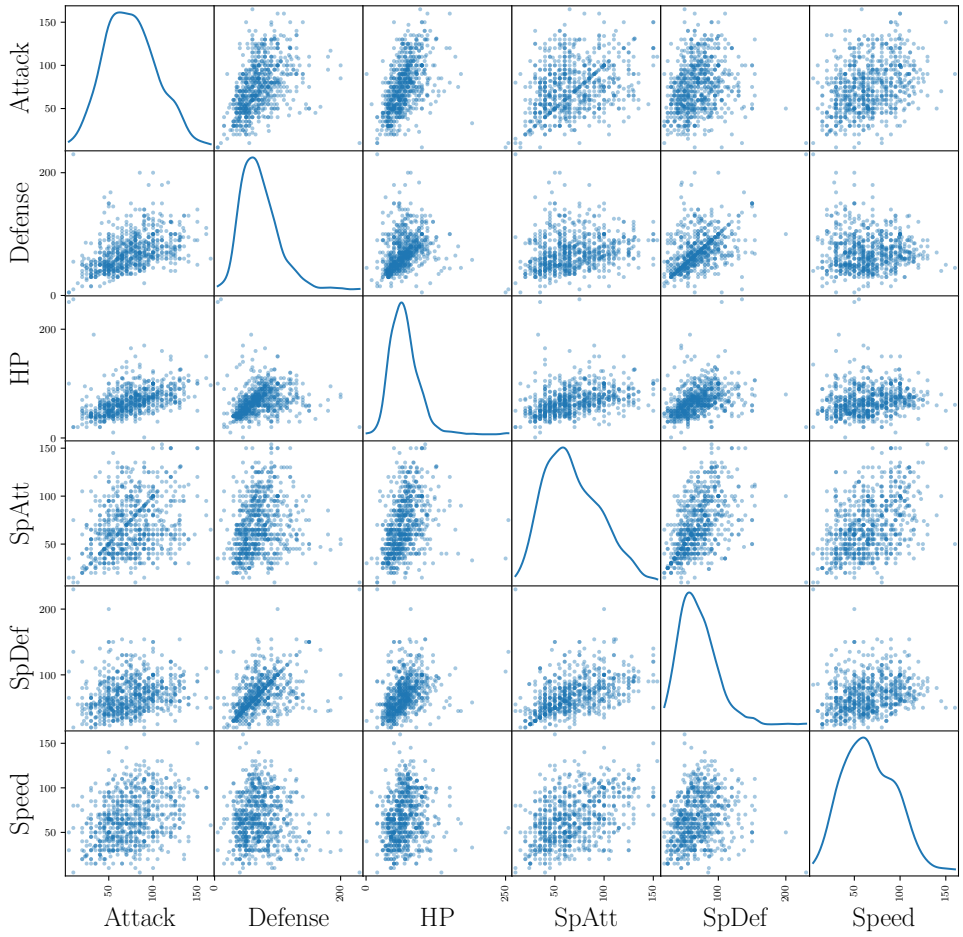


Figure 7.4: *Pairs-plot of the Pokémon six fight characteristics*

folding test of unimodality can help the clustering parametrization step.

Catching the behavior jumps In this experiment we analyze a part of the DJIA 30 Stock Time Series dataset hosted on `kaggle`[2]. In particular we study the historical stock data from NYSE (between 2006-01-03 to 2017-29-12) of four major companies: Cisco (CSCO), Intel (INTC), IBM and Microsoft (MSFT). We consider the highest price daily reached, so we get a dataset with $n = 3018$ observations in dimension 4 (number of companies).

[2] `https://www.kaggle.com/szrlee/stock-time-series-20050101-to-20171231`

Method	Nb of clusters	Avg. silhouette
GMM (with AIC)	≥ 5	-0.0045
GMM (with BIC)	4	0.026
Gap statistic	2	0.287
Mean shift	3	0.55
Affinity propagation	2	0.181

Table 7.2: *Number of clusters and average silhouette score according to the clustering method*

The figure 7.5 shows the evolution of the stock prices for these four companies. We may notice two different behaviors: IBM stocks vary more than Cisco, Intel and Microsoft ones. Now, we compute the folding statistics Φ over a sliding window of size

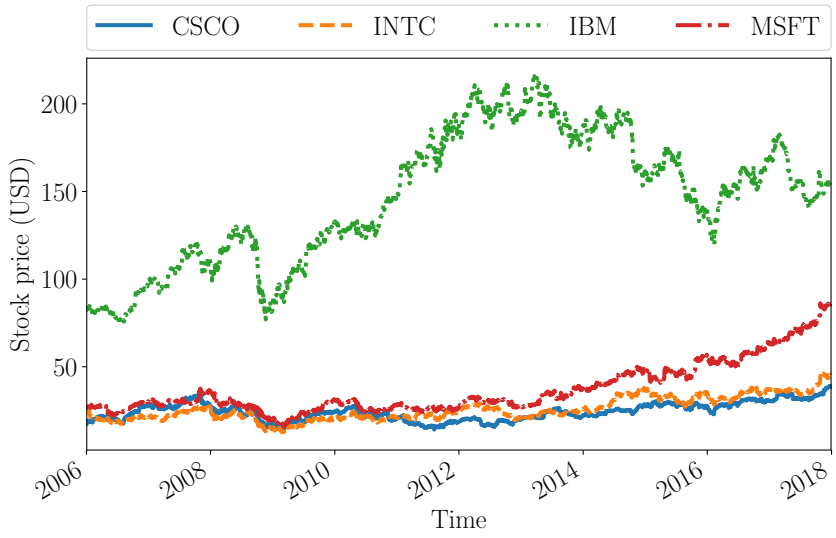


Figure 7.5: *Stock evolution of the four companies*

650 (about two years and half, because there is no data on Saturday and Sunday). It means that at every iteration, we take a snapshot of 650 observations in dimension 4, and we compute Φ on it. The results are presented in the figure 7.6. We add significance bounds at level 0.05 ($q \simeq 0.15$). One can roughly see three "moments of unimodality" (around iterations 500, 1500 and 1800).

What does it mean? If the distribution at these moments is unimodal, it means that the companies stocks are individually quite the same: their behavior is stationary within the window. At the moments when the distribution is multimodal, some stock prices

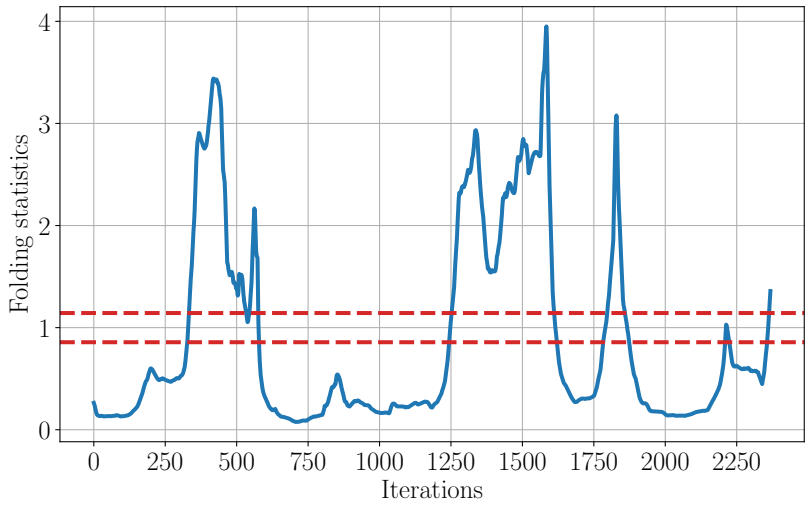


Figure 7.6: *Folding statistics within 650 days time windows*

are jumping.

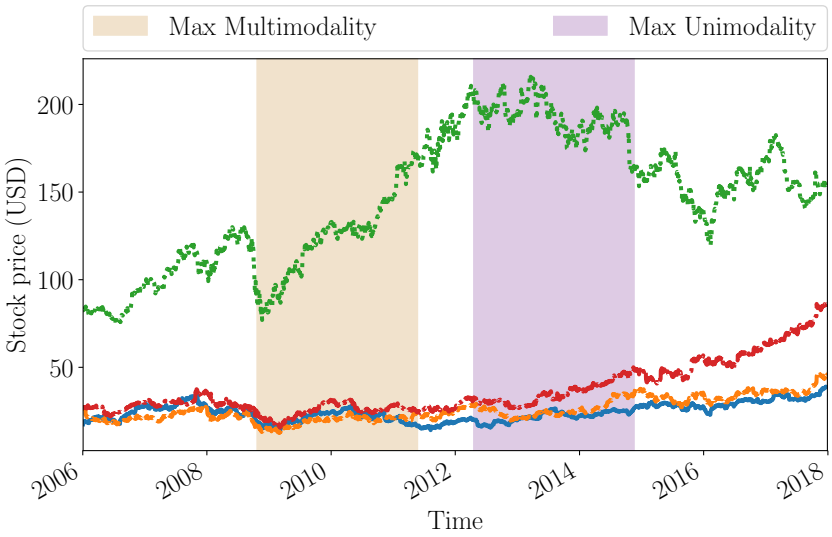
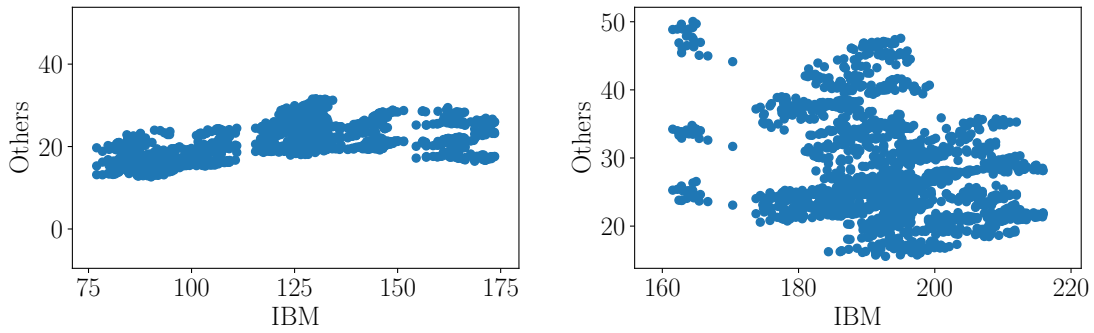


Figure 7.7: *Extreme time windows*

To highlight this phenomenon, we show two windows on the time series (figure 7.7). One corresponding to the minimum of Φ (*maximum of multimodality*) and the other one corresponding to its maximum (*maximum of unimodality*). Indeed, we may notice that the most multimodal window is when the IBM stocks jump a lot unlike the others although the most unimodal window occurs when the four stocks are the most stationary.

If we look deeper at the scatter plots within these two windows (figure 7.8), we can confirm these behaviors (to ease the plot we have overlaid the scatter-plots IBM vs MSFT, IBM vs CSCO and IBM vs INTC). On figure 7.8a, we can observe about three clusters whereas no clusters really arises on the figure 7.8b. However we clearly see three little groups of data on this latter figure. In fact, they are not statistically substantial (not real modes).



(a) *Maximum of multimodality* **(b)** *Maximum of unimodality*

Figure 7.8: *Scatter-plots IBM vs Others*

Running time We compare the running time of two methods: computing the statistics on the complete window each time (batch computation), or using the incremental approach presented in section 6.3. The complete computation (2400 iterations) took 5.5 seconds with batch computation, and 4.5 seconds with the incremental algorithm (around 18% acceleration). Note that the variance computation part is identical in both methods: if we only compare the computation of s^*, the times are 3.4 and 0.9 seconds, giving a 73% decrease on running time. We can conclude that even on this relatively simple data (low dimension, few iterations), the incremental computations has a significant impact on running time.

Helping clustering algorithms We have mentioned in the Pokémon experiment (section 7.4) that unimodality testing provides a different piece of information from clustering. In that case, it was relevant to avoid over-clustering. Here, we use it as a parametrization support for clustering.

Let us consider the DBSCAN algorithm [31]. It requires two parameters: the neighborhood radius r and the minimum requested numbers of neighbors to be considered as a core point m_s. Our idea is to tune the radius parameter so as to make DBSCAN output "1 cluster" only when $\Phi > 1$. Obviously, we cannot ensure the quality of the clustering in the other regions however with this criterion we can narrow down the parameter research domain.

An example with $m_s = 80$ (12.5% of the observations in the window) is presented on the figure 7.9. We plot the folding statistics (top) and the number of clusters output by DBSCAN with different neighborhood radius ($r = 6, 7, 8, 9$ and 10). Moreover we add all the unimodal areas (where $\Phi > 1 + q$) and the first multimodal area (where $\Phi < 1 - q$).

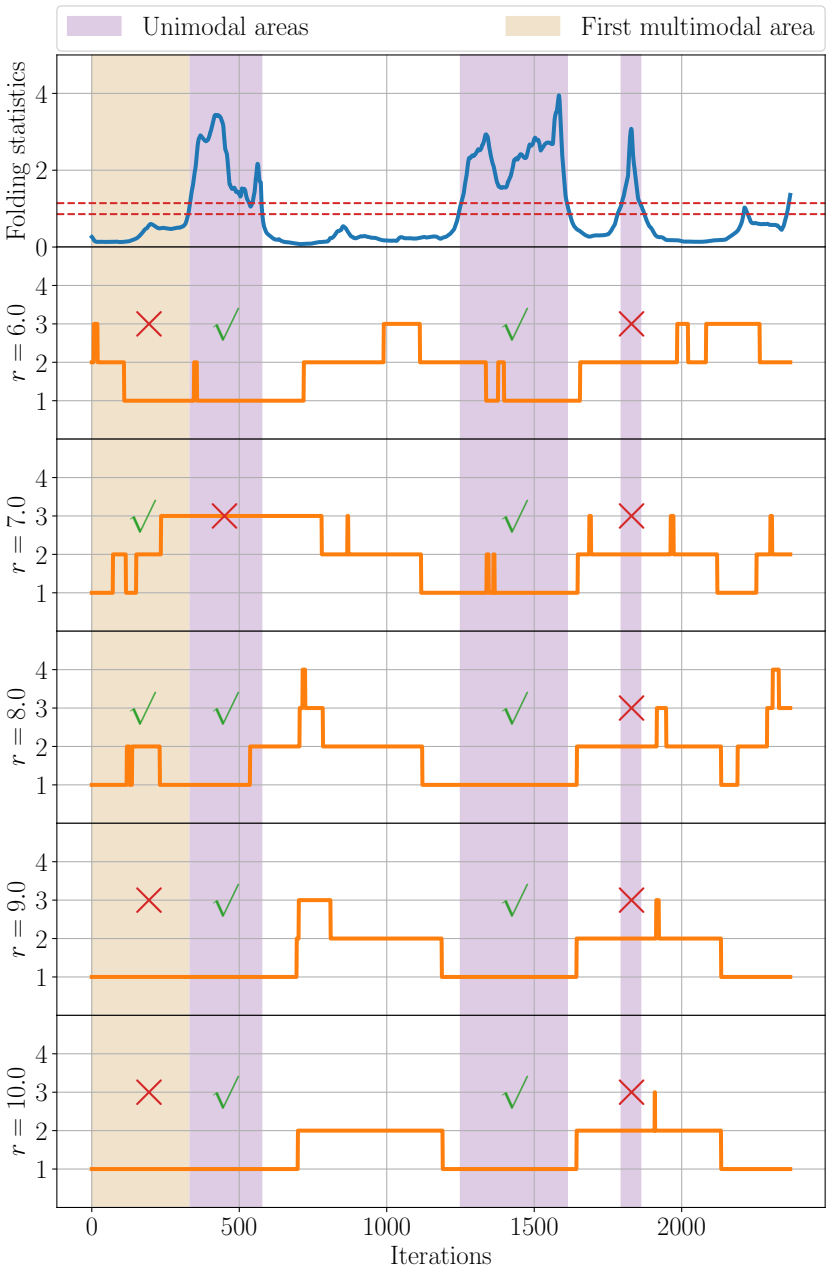


Figure 7.9: *Number of clusters output by DBSCAN according to the radius r*

Let us analyze the unimodal area around iteration 1500. All the instances of DBSCAN output nearly a single cluster. But around it. 1800 (unimodal area too), they all return 2 clusters. In the first unimodal zone (around it. 500) DBSCAN($r = 7$) is the only one not returning one cluster. Eventually, if we look at the first multimodal area (before it. 330), only the instances DBSCAN($r = 7$) and DBSCAN($r = 8$) output several clusters. We may extend the analysis to all the regions, making the instance DBSCAN($r = 8$) the more faithful to the folding test of unimodality.

We insist on the non-completeness of our test. We cannot claim that DBSCAN gives the correct clusters in the multimodal regions. Moreover, we know that other phenomenons should be taken into account for clustering like the influence of the second parameter and/or data normalization. We merely show that our lightweight test is able to provide some paramount information about data. The latter may be useful for a potential clustering stage.

7.5 Conclusion

This chapter introduces the folding test of unimodality (FTU). To the best of our knowledge, this is the first time a multivariate and purely statistical unimodality test has been proposed. The test is based on a new folding statistic Φ, which provides a score related to the "level of unimodality" of a distribution. This statistic is easy and fast to compute and does not require to estimate the underlying cdf or density. Furthermore Φ does not depend on the parameters of the input distribution and its computation can even be adapted to streaming contexts. The only parameter of FTU is a natural $p-$value giving the desired significance level of the result.

From both the theoretical and the practical sides, we experiment the relevance and the efficiency of FTU. Its properties may lead to use it at in different levels: from a simple descriptive statistics to a core component of a more complex algorithm.

In the next part we will develop another generalization of FTU, called HFTU (Hyperplane Folding Test of Unimodality). This latter test does not hide the presented results as it actually embeds different properties useful in certain contexts. This test will notably be in the core of a new clustering algorithm ($\Phi-$means).

8

HFTU: Hyperplane Folding Test of Unimodality

Unimodality testing remains a restricted research topic despite its closeness to clustering. As we have seen in the previous part, it is confined to univariate distributions, limiting the scope of application.

For this purpose, we proposed the Folding Test of Unimodality (FTU) [116] which is the first multivariate and purely statistical unimodality test. It is then able to estimate the unimodality (or the multimodality) of a multivariate dataset without any dimension reduction trick. In addition to its dimensional scalability, FTU has other advantages: its complexity linearly depends on the number of observations, it can be computed/updated incrementally over a sliding window and finally it is rather easy to implement.

FTU, in the univariate case, relies on the interplay between symmetry and variance. This idea proved to be very powerful. However, the generalization to the multivariate case introduces some limits and drawbacks. The main limitation is that the underlying

optimization problem may not have a solution. This requires to use approximations, which may cause FTU to fail when facing certain multimodal symmetrical distributions.

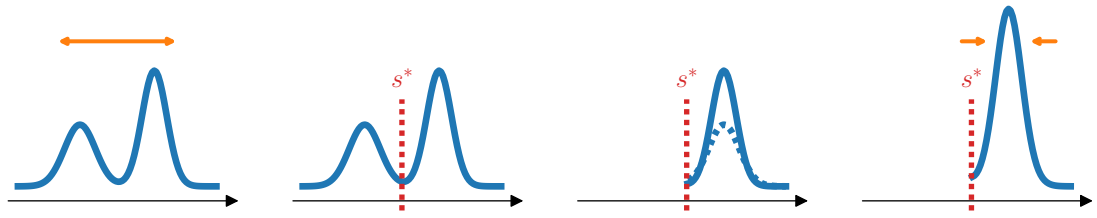
Here we detail the Hyperplane Folding Test of Unimodality (HFTU) which is another generalization of the univariate folding test of unimodality to higher dimensions. HFTU provides more theoretical guarantees making it more robust than FTU against data distribution. In addition, HFTU builds an hyperplane $\mathcal{H}^*$ which provides a global description of the distribution underlying data. Such a hyperplane will be a key element to build a clustering algorithm.

In this part we may use some notations already introduced in the FTU description. Without ambiguity, we will use the subscript "FTU" to clearly denote these mentions. Some proofs are not presented in this chapter but can be found in the appendix B.

8.1 Back to the folding mechanism

Here we merely recall the idea developed in 6 as it is fundamental to describe the new generalization.

Understanding FTU in dimension 1. In bimodal density (figure 8.1), the variance is likely to be "large" because the two modes make the data far from the expected value (stage (a)). The idea is the following: find a good pivot s^* (stage (b)) and fold a mode onto the other (stage (c)). The resulting density will then have a far lower variance (stage (d)). Intuitively, this phenomenon will not appear for unimodal distributions (actually not with the same amplitude).



(a) *Initial distribution* **(b)** *Finding the right pivot* **(c)** *Folding about the pivot* **(d)** *Folded distribution*

Figure 8.1: *Folding mechanism*

If we assume getting the pivot s^*, the folding step (stages (a), (b) and (c)) is performed with the transformation $X \mapsto |X - s^*|$. The impact of this mechanism is evaluated through the folding ratio:

$$\phi_{\text{FTU}}(X) = \frac{\text{Var}\,|X - s^*|}{\text{Var}\,X}. \quad \begin{array}{l} \leftarrow \textit{folded variance} \\ \leftarrow \textit{initial variance} \end{array}$$

Finally, the test itself compares $\phi_{\mathrm{FTU}}(X)$ and $\phi_{\mathrm{FTU}}(U)$ with U uniformly distributed (the range having no impact) through the folding statistics $\Phi_{\mathrm{FTU}}(X) = \phi_{\mathrm{FTU}}(X)/\phi_{\mathrm{FTU}}(U)$. A distribution is then considered as multimodal once we can fold it *better* than the uniform distribution, i.e. when $\Phi_{\mathrm{FTU}}(X) < 1$ (otherwise it is considered as unimodal). The uniform case is a common unimodal distribution limit (see [53, 116] for example) and we will give some details in 8.4.

How FTU is generalized. The extension to the multivariate case seems natural whereas the representation of the folding mechanism is unfortunately less intuitive. The folding step is then performed through the transformation $X \mapsto \|X - s^*\|, s^* \in \mathbb{R}^d$ and the "initial variance" is summed up through the trace of the covariance matrix, so the generalization of the folding ratio is the following:

$$\phi_{\mathrm{FTU}}(X) = \frac{\operatorname{Var}\|X - s^*\|}{\operatorname{Tr}\Sigma}. \tag{8.1}$$

Eventually, the folding statistics $\Phi_{\mathrm{FTU}}(X)$ uses the uniform $d-$ball as the reference distribution.

Issues around the folding pivot. The first feature to notice is that FTU performs a *central symmetry folding* about the pivot s^*. The obvious aim is to find the best pivot, i.e. the one which reduces the variance the most: $\operatorname{argmin}_{s\in\mathbb{R}^d} \operatorname{Var}\|X - s\|$. However, depending on the distribution of X, the function $s \mapsto \operatorname{Var}\|X - s\|$ has not necessary a minimum (precisely in the multivariate case). To avoid this problem, we need to use an approximation, defining $s^* = \operatorname{argmin}_{s\in\mathbb{R}^d} \operatorname{Var}\left(\|X - s\|^2\right)$ which has a closed solution $s^* = (1/2)\cdot\Sigma^{-1}\cdot\operatorname{Cov}\left(X, \|X\|^2\right)$ once $\mathbb{E}(X^4)$ exists. Unfortunately this approximation can make FTU fail (see example 8.1 below).

EXAMPLE 8.1 *Let $a > 0$ and let us consider the random variable X such that $\mathbb{P}(X = -a) = \mathbb{P}(X = a) = \mathbb{P}(X = 0) = 1/3$. X is clearly multimodal and we can show that $s^* = 0$ while $\operatorname{argmin}_{s\in\mathbb{R}} \operatorname{Var}|X - s| = \pm a/4$. This difference between the best pivot and the approximation leads to $\Phi_{\mathrm{FTU}}(X) = 4/3 > 1$, so FTU considers the distribution of X as unimodal. By using the best pivot $\pm a/4$, the folding statistic is 1 (which is actually a limit case).*

Proof. First we give basic moments of X:

$$\mathbb{E}(X) = 0 \quad \text{and} \quad \mathbb{E}(X^2) = \frac{2}{3}a^2.$$

Without loss of generality, let us assume $0 \le s \le a$ (when $|s| > a$, we have $\operatorname{Var}|X - s| = \operatorname{Var} X$, so actually it does not reduce the variance). The distribution of $|X - s|$ is the following:

$\lvert X - s\rvert$	$a + s$	s	$a - s$
$\mathbb{P}\lvert X - s\rvert$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

So its expected value is $\mathbb{E}\,|X-s| = \frac{2}{3}a + \frac{1}{3}s$. Moreover

$$\begin{aligned}
\operatorname{Var}|X-s| &= \mathbb{E}\left(|X-s|^2\right) - \left(\mathbb{E}\,|X-s|\right)^2 \\
&= \mathbb{E}(X^2 - 2sX + s^2) - \left(\frac{2}{3}a + \frac{1}{3}s\right)^2 \\
&= \frac{2}{3}a^2 + s^2 - \left(\frac{2}{3}\right)^2 a^2 - \left(\frac{1}{3}\right)^2 s^2 - \frac{4}{9}as \\
&= \frac{2}{9}\left(a^2 + 4s^2 - 2as\right).
\end{aligned}$$

The expression of $\operatorname{Var}|X-s|$ corresponds to a convex parabola. Its is minimum when

$$\frac{\mathrm{d}}{\mathrm{d}s}\operatorname{Var}|X-s^*| = 0 \quad \Longleftrightarrow \quad s^* = \frac{1}{4}a.$$

Assuming $-a \leq s \leq 0$, we get $s^* = -\frac{1}{4}a$. □

The FTU generalization leads to use an approximation which is not tight and makes the test fail for some symmetric distributions.

Design of HFTU. The generalization of FTU seems to go against the original idea described in figure 8.1 where we are likely to consider s^* as a symmetry axis. As another generalization, we exploit this idea by providing a unimodality test based on *axial symmetry folding*, where axis naturally corresponds to an hyperplane when the dimension d is higher than 1 (see figure 8.2).

The main advantage of this choice is that the hyperplane $\mathcal{H}^*$ which folds the best always exists once X is twice integrable (existence theorem developed in section 8.3). By computing $\mathcal{H}^*$, it allows us to build a unimodality test (HFTU) we detail in section 8.4. Furthermore, $\mathcal{H}^*$ has several robustness assets that we will experiment in the last section.

8.2 Problem statement

In this section we show how the folding is performed by defining the problem and proving the existence of a solution (some missing proofs can be found in appendix B).

Our goal is to find an affine hyperplane $\mathcal{H}^*$ such that the variance of the folded data about $\mathcal{H}^*$ is the lowest possible. First we introduce some definitions and then we show that such a minimum exists.

An affine hyperplane is basically represented by a couple $(u, u_0) \in \mathbb{R}^d \setminus \{0\} \times \mathbb{R}$ where u is a normal vector of $\mathcal{H}$ and u_0 is the intercept. Without loss of generality we can restrict to the case $\|u\| = 1$ (unit vector), because if u denotes a normal vector of an hyperplane $\mathcal{H}$, $k \cdot u$ ($k \in \mathbb{R}$) is also a normal vector of $\mathcal{H}$.

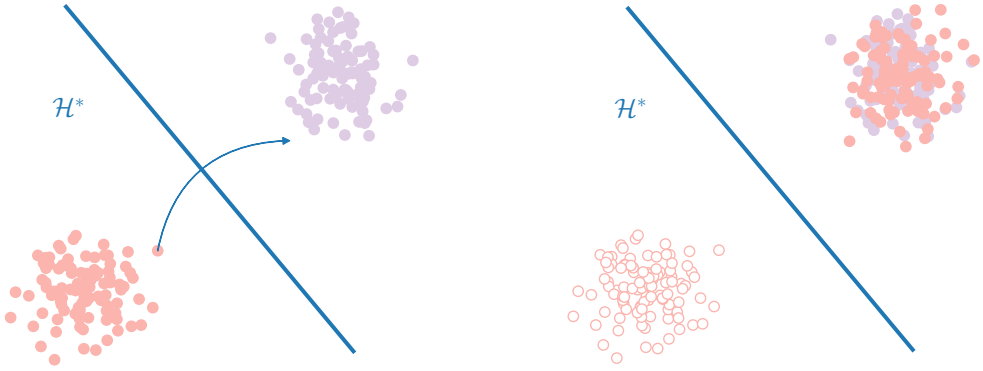


Figure 8.2: *HFTU folding. Rather than folding about a point, HFTU maps all the data to a single side of an hyperplane $\mathcal{H}^*$ so as to minimize the overall variance.*

DEFINITION 8.1 *Let $u \in \mathcal{S}^d, u_0 \in \mathbb{R}$ and $\lambda = (u, u_0)$. We note $\mathcal{H}_\lambda$ the hyperplane of $\mathbb{R}^d$ defined by*

$$\mathcal{H}_\lambda = \left\{ x \in \mathbb{R}^d \;\middle|\; h_\lambda(x) = u^T x + u_0 = 0 \right\}.$$

Moreover, the orthogonal projection on $\mathcal{H}_\lambda$ is given by $p_\lambda : x \mapsto x - h_\lambda(x)u$. Finally the distance between $x \in \mathbb{R}^d$ and $\mathcal{H}_\lambda$ is $d(x, \mathcal{H}_\lambda) = |h_\lambda(x)|$.

Now we define the *folding function*: when an hyperplane is defined, some data are "above" it (in the direction of u) although the others are "below" it (in the direction of $-u$). The folding maps all the data above the hyperplane while keeping the distances from it (see also figure 8.2).

DEFINITION 8.2 *Let $\lambda = (u, u_0) \in \mathcal{S}^d \times \mathbb{R}$ and $\mathcal{H}_\lambda$ the corresponding hyperplane. The* ***folding function*** Fold_λ *is defined as*

$$\mathrm{Fold}_\lambda : \begin{array}{ccc} \mathbb{R}^d & \to & \mathbb{R}^d \\ x & \mapsto & p_\lambda(x) + d(x, \mathcal{H}_\lambda)u \end{array}$$

Moreover, if X is a twice integrable random vector of $\mathbb{R}^d$, we define its ***folded variance*** *about $\mathcal{H}_\lambda$ as $V_\lambda(X) = \mathrm{Tr}\,\mathrm{Cov}\,\mathrm{Fold}_\lambda(X)$.*

The folding mechanism is quantified by the *folded variance* which is roughly the amount of variance in the folded dataset (the sum of variance in every direction). The next proposition gives an explicit formula which is found in developing $\mathrm{Tr}\,\mathrm{Cov}\,\mathrm{Fold}_\lambda(X)$.

PROPOSITION 8.1 *Let $\lambda = (u, u_0) \in \mathcal{S}^d \times \mathbb{R}$. Let X be a twice integrable random vector of $\mathbb{R}^d$ with covariance matrix Σ. Then, the expression of the folded variance of X w.r.t. $\mathcal{H}_\lambda$ is given by:*

$$V_\lambda(X) = \operatorname{Tr}\Sigma + \mathbb{E}\left(h_\lambda(X)\right)^2 - \mathbb{E}\left(|h_\lambda(X)|\right)^2. \tag{8.2}$$

Proof. First, let us write $\operatorname{Fold}_\lambda(X)$:

$$\begin{aligned}\operatorname{Fold}_\lambda(X) &= p_\lambda(X) + d(X, \mathcal{H}_\lambda)u \\ &= X - \left(h_\lambda(X) - |h_\lambda(X)|\right)u\end{aligned}$$

Therefore the $i-$th component of $\operatorname{Fold}_\lambda(X)$, noted $(\operatorname{Fold}_\lambda(X))_i$ has the following variance:

$$\operatorname{Var}\left[\operatorname{Fold}_\lambda(X)_i\right] = \operatorname{Var} X_i + u_i^2 \operatorname{Var}\left(|h_\lambda(X)| - h_\lambda(X)\right) + 2\operatorname{Cov}\left(u_i X_i, |h_\lambda(X)| - h_\lambda(X)\right).$$

The folding variance is then given by:

$$\begin{aligned}\operatorname{Tr}\operatorname{Cov}\operatorname{Fold}_\lambda(X) &= \operatorname{Tr}\Sigma + \operatorname{Var}\left(|h_\lambda(X)| - h_\lambda(X)\right)\sum_{i=1}^d u_i^2 \\ &\quad + 2\operatorname{Cov}\left(\sum_{i=1}^d u_i X_i, |h_\lambda(X)| - h_\lambda(X)\right) \\ &= \operatorname{Tr}\Sigma + \operatorname{Var}\left(|h_\lambda(X)| - h_\lambda(X)\right) + 2\operatorname{Cov}\left(h_\lambda(X), |h_\lambda(X)| - h_\lambda(X)\right).\end{aligned}$$

But we have

$$\begin{aligned}\operatorname{Var}\left(|h_\lambda(X)| - h_\lambda(X)\right) &= \operatorname{Var}|h_\lambda(X)| + \operatorname{Var} h_\lambda(X) - 2\operatorname{Cov}\left(|h_\lambda(X)|, h_\lambda(X)\right) \\ 2\operatorname{Cov}\left(h_\lambda(X), |h_\lambda(X)| - h_\lambda(X)\right) &= 2\operatorname{Cov}\left(h_\lambda(X), |h_\lambda(X)|\right) - 2\operatorname{Var} h_\lambda(X).\end{aligned}$$

So we have:

$$\operatorname{Tr}\operatorname{Cov}\operatorname{Fold}_\lambda(X) = \operatorname{Tr}\Sigma + \operatorname{Var}|h_\lambda(X)| - \operatorname{Var} h_\lambda(X).$$

Now, we can rewrite the variances:

$$\begin{aligned}\operatorname{Var}|h_\lambda(X)| &= \mathbb{E}\left(h_\lambda^2(X)\right) - \mathbb{E}\left(|h_\lambda(X)|\right)^2 \\ &= \operatorname{Var} h_\lambda(X) + \mathbb{E}\left(h_\lambda(X)\right)^2 - \mathbb{E}\left(|h_\lambda(X)|\right)^2.\end{aligned}$$

So,

$$\operatorname{Var}|h_\lambda(X)| - \operatorname{Var} h_\lambda(X) = \mathbb{E}\left(h_\lambda(X)\right)^2 - \mathbb{E}\left(|h_\lambda(X)|\right)^2$$

and finally,

$$V_\lambda(X) = \operatorname{Tr}\Sigma + \mathbb{E}\left(h_\lambda(X)\right)^2 - \mathbb{E}\left(|h_\lambda(X)|\right)^2.$$

□

Given an hyperplane (a couple (u, u_0)), we are able to compute how much the variance is reduced when we fold data about it. Our goal is to find the "best" hyperplane, the one which reduces the folded variance the most.

PROBLEM 8.1 *Let X be a twice integrable random variable with covariance matrix Σ. HFTU solves the following optimization problem:*

$$\min_{\lambda=(u,u_0)\in\mathcal{S}^d\times\mathbb{R}} \operatorname{Tr}\Sigma + \mathbb{E}\left(h_\lambda(X)\right)^2 - \mathbb{E}\left(|h_\lambda(X)|\right)^2. \tag{8.3}$$

We keep the constant $\operatorname{Tr}\Sigma$ in the optimization problem not to break the link with V_λ. At this point, we have not any clues about this function $\lambda \mapsto V_\lambda(X)$. The next section aims to give theoretical results showing that such a minimum exists and can then be reached.

REMARK 8.1 *In dimension $d = 1$ the problem (8.3) is equivalent to solve $\min_{s\in\mathbb{R}} \operatorname{Var}|X - s|$. It means that HFTU puts the initial idea of FTU into practice, while the generalization of the latter coerces into using an approximation.*

8.3 Existence theorem

Unfortunately, the problem (8.3) has not the usually expected properties: it is not convex and rather hard to simplify because of the absolute value. In this paragraph, we nevertheless show that the problem (8.3) does have a solution. It makes a real difference with the classical FTU which has not such a property and need then to find an approximation. First we give properties of $\lambda \mapsto V_\lambda(X)$.

LEMMA 8.1 *Let X be a twice integrable random vector of $\mathbb{R}^d$. Then the function $\lambda \mapsto V_\lambda(X)$ is continuous and bounded on $\mathcal{S}^d \times \mathbb{R}$. In particular, for all $\lambda \in \mathcal{S}^d \times \mathbb{R}, 0 \leq V_\lambda(X) \leq \operatorname{Tr}\Sigma$.*

Proof. The function $v : \mathcal{S}^d \times \mathbb{R} \to \mathbb{R}, \lambda \mapsto \operatorname{Tr}\Sigma + \mathbb{E}\left(h_\lambda(X)\right)^2 - \mathbb{E}\left(|h_\lambda(X)|\right)^2$ is a composition of continuous functions, so it is continuous. The function v is also positive on $\mathbb{R}$ as it is a sum of variances. But we can also notice that it is upper bounded by $\operatorname{Tr}\Sigma$. Indeed for all $\lambda \in \mathcal{S}^d \times \mathbb{R}, |\mathbb{E}\left(h_\lambda(X)\right)| \leq \mathbb{E}\left(|h_\lambda(X)|\right)$ thanks to the Jensen's inequality, so $\mathbb{E}\left(h_\lambda(X)\right)^2 \leq \mathbb{E}\left(|h_\lambda(X)|\right)^2$ and $v(\lambda) \leq \operatorname{Tr}\Sigma$. □

From the lemma 8.1, we cannot directly say that $\lambda \mapsto V_\lambda(X)$ will reach its infimum because the definition domain $\mathcal{S}^d \times \mathbb{R}$ is not compact. The problem naturally comes from u_0 which is not bounded. To circumvent this hindrance, we need to show that for all unit normal vector u the application $t \mapsto V_{(u,t)}(X)$ reaches a minimum.

LEMMA 8.2 *Let X be a random vector of $\mathbb{R}^d$ such that $\mathbb{E}(X)$ exists. Let $u \in \mathbb{R}^d$, then we have the following second order expansion:*

$$\mathbb{E}\left|1 + \alpha \cdot u^T X\right| \underset{\alpha\to 0}{=} 1 + \alpha \cdot u^T \mathbb{E}(X) + o(\alpha^2).$$

The lemma 8.2 allows notably to prove that for all direction u, $V_{(u,t)}(X)$ tends to $\operatorname{Tr}\Sigma$ when t is large ($\pm\infty$). This result is quite natural because if we fold about a plane very distant from the high density regions, the folded distribution will be very close to the initial one, so its variance. To a certain extent, we could also use the notation $V_\infty(X)$ to denote $\operatorname{Tr}\Sigma$. The lemma 8.3 uses the lemma 8.2 to show that in every direction u, we can find an intercept which minimizes the folded variance.

LEMMA 8.3 *Let $u \in \mathcal{S}^d$ and let X be a twice integrable random vector of $\mathbb{R}^d$. Then the univariate function $t \mapsto V_{(u,t)}(X)$ has an absolute minimum on $\mathbb{R}$.*

Proof. Let $u \in \mathcal{S}^d$. Let us note $v : \mathbb{R} \to \mathbb{R}, t \mapsto V_{(u,t)}(X)$. With lemma 8.2, we can develop v when $t \to \pm\infty$.

$$\begin{aligned}
v(t) &= \operatorname{Tr}\Sigma + \left(\mathbb{E}(u^T X + t)\right)^2 - \left(\mathbb{E}\left|u^T X + t\right|\right)^2 \\
&= \operatorname{Tr}\Sigma + t^2\left(1 + \frac{\mathbb{E}(u^T X)}{t}\right)^2 - t^2\left(\mathbb{E}\left|1 + \frac{u^T X}{t}\right|\right)^2 \\
&\underset{t\to\pm\infty}{=} \operatorname{Tr}\Sigma + t^2\left(1 + \frac{\mathbb{E}(u^T X)}{t}\right)^2 - t^2\left(1 + \frac{\mathbb{E}(u^T X)}{t} + o\left(\frac{1}{t^2}\right)\right)^2 \\
&\underset{t\to\pm\infty}{=} \operatorname{Tr}\Sigma + o(1)
\end{aligned}$$

The lemma 8.1 told us that v is also continuous and bounded between 0 and $\operatorname{Tr}\Sigma$. Let $\epsilon > 0$, then $t_0 > 0$ exists such that for all $|t| > t_0, \operatorname{Tr}\Sigma - \epsilon \leq v(t) \leq \operatorname{Tr}\Sigma$. The restricted function $v_{|[-t_0,t_0]}$ is then continuous on a bounded domain so it reaches its bounds. Its lower-bound is in fact those of v. □

Therefore, we have elements showing that the best hyperplane will be where the data are, so we could indeed reach it. Theorem 8.1 finally gives this desired result. In the next sections, we will use $V_{\mathcal{H}^*}(X)$ to denotes this absolute minimum.

THEOREM 8.1 *Let X be a twice integrable random vector of $\mathbb{R}^d$. Then the application $\lambda \mapsto V_\lambda(X)$ reaches an absolute minimum on $\mathcal{S}^d \times \mathbb{R}$.*

Proof. Let us introduce $f : \mathcal{S}^d \times \mathbb{R} \to \mathbb{R}, \lambda \mapsto V_\lambda(X)$. This function f is bounded on $\mathcal{S}^d \times \mathbb{R}$, so let us note $I = \inf_{\lambda \in \mathcal{S}^d \times \mathbb{R}} f(\lambda)$ which is finite. We can pick-up a sequence $(f_n)_{n\in\mathbb{N}}$ of $\operatorname{Im}(f)$ converging to I. At each f_n corresponds a tuple (u_n, t_n), we then merely have $f(u_n, t_n) \to I$.

$(u_n)_{n\in\mathbb{N}}$ is a sequence of the compact set $\mathcal{S}^d$, so we can extract a convergent subsequence $(u_{\psi(n)})_{n\in\mathbb{N}}$ (Bolzano-Weierstrass theorem). Let us note $u \in \mathcal{S}^d$ its limit. As f is continuous, for all $t \in \mathbb{R}, f(u_{\psi(n)}, t) \to f(u, t)$.

Thus we have the equivalence $f_{\psi(n)} = f(u_{\psi(n)}, t_{\psi(n)}) \sim f(u, t_{\psi(n)})_{n\in\mathbb{N}}$. But, the limit of $(f_{\psi(n)})$ is unique and equal to I, so it implies $f(u, t_{\psi(n)}) \to I$. With lemma 8.3, we know that the function $t \mapsto f(u, t)$ has a minimum, we note $f_{\min}(u)$. Then for all $n \in \mathbb{N}$, $f(u, t_{\psi(n)}) \geq f_{\min}(u)$, and finally $I \geq f_{\min}(u)$. The definition of I naturally implies $I \leq f_{\min}(u)$, so finally we have the equality showing that this infimum is reached. □

8.4 The Hyperplane folding test of unimodality

We have seen in the previous section, that under low restrictive condition on the distribution of X, we can find a hyperplane which cuts down the variance when X is folded about it. To make a decision about the unimodality character of X we need two things: to quantify the variance reduction and to define a reference to compare with. It will lead to define the HFTU test.

Quantifying the folding mechanism The natural way to estimate the folding step is to compare the folded variance with the initial one. It leads then to introduce the folding ratio ϕ, as it is made in the original FTU (see expression (8.1)).

DEFINITION 8.3 *Let X be a twice integrable random vector of $\mathbb{R}^d$. We define its* ***folding ratio*** *$\phi(X)$ as the ratio between the minimum of folded variance and its initial variance:*

$$\phi(X) = \frac{V_{\mathcal{H}^*}(X)}{\operatorname{Tr}\Sigma} = \frac{V_{\mathcal{H}^*}(X)}{V_\infty(X)}.$$

According to the distribution, the efficiency of the folding is not the same. Our test relies on the fact that the variance is far more reduced for multimodal distributions than for unimodal ones. When the variance is significantly cut down (multimodal case), the folding ratio $\phi(X)$ is then *low* although $\phi(X)$ is quite *high* when the folding is not efficient (unimodal case).

Thus, we need a reference to discriminate unimodal case from multimodal case. The uniform density is commonly considered as the *worst case of unimodality* ([53, 116]). In a mathematical point of view, unimodal densities shape a convex set where uniform densities are locating at its edges (theorem 1.2 in [24]). So the uniform case is a natural reference to compare unimodal densities from multimodal ones. The proposition 8.2 gives the folding quantities of the uniform $d-$ball. We will use these results to build a general statistics to separate unimodal distributions from multimodal ones.

PROPOSITION 8.2 *Let $r > 0$. Let U_d be random vector of $\mathbb{R}^d$ uniformly distributed over the $d-$ball of radius r. The minimum of the folded variance $V_{(u,u_0)}(U_d)$ is reached at $u_0 = 0$, and is given by:*

$$V_{\mathcal{H}^*}(U_d) = r^2 \cdot \left(\frac{d}{d+2} - \left(\frac{a_d}{d+1} \right)^2 \right).$$

Furthermore, the folding ratio ϕ_d of U_d does not depend on r and is given by

$$\phi_d = \phi(U_d) = 1 - \frac{(d+2)a_d^2}{d(d+1)^2} \quad \text{with} \quad a_d = \frac{2}{\sqrt{\pi}} \frac{\Gamma\left(\frac{d}{2}+1\right)}{\Gamma\left(\frac{d}{2}+\frac{1}{2}\right)} = \begin{cases} \dfrac{4^p}{\pi} \dfrac{p!(p-1)!}{(2p-1)!} & \text{when } d = 2p \\ \dfrac{(2p+1)!}{4^p (p!)^2} & \text{when } d = 2p+1 \end{cases}$$

HFTU Let us introduce the hyperplane folding test of unimodality (HFTU). For any distribution having a variance, we can compute the efficiency of the folding stage (through the folding ratio ϕ). Now we claim that this folding ratio will be lower than ϕ_d when the distribution is unimodal and it will be higher than ϕ_d it in the multimodal case. So we define the folding statistic (definition 8.4) and HFTU (definition 8.5) below.

DEFINITION 8.4 *Let X be a twice integrable random vector of $\mathbb{R}^d$. We define its* ***folding statistics*** *$\Phi(X)$ as the ratio between its folding ratio and the uniform $d-$ball ones:*

$$\Phi(X) = \frac{\phi(X)}{\phi_d}$$

DEFINITION 8.5 (HFTU: Hyperplane Folding Test of Unimodality) *Let X be a twice integrable random vector of $\mathbb{R}^d$.*

$$\begin{aligned}
&\Phi(X) > 1 \Longrightarrow X \text{ unimodal} \\
&\Phi(X) < 1 \Longrightarrow X \text{ multimodal} \\
&\Phi(X) = 1 \Longrightarrow \text{undefined } (X \text{ is a limit case})
\end{aligned}$$

The case $\Phi(X) = 1$ remains for us a theoretical limit case where the decision cannot be made. By design, we have $\Phi(U) = 1$ for U drawn uniformly within the $d-$ball (unimodal) but we also have $\Phi(X) = 1$ in the configuration described in the example 8.1 (multimodal). In practice, significance levels are used to avoid the undefined case (see §8.5).

8.5 Computational aspects and test significance

In this section we show a way to compute the folding statistics $\Phi(X)$. Moreover, we give a model to compute $p-$values aimed to estimate the significance of the test.

Computing the folding statistics $\Phi(X)$ To perform HFTU, we mainly need to solve the optimization problem (8.3). In practice, we have a finite sample of i.i.d. observations $X^{(1)}, X^{(2)}, \dots X^{(n)} \in \mathbb{R}^d$. In this case, integral operators like the expected value or the covariance are merely replaced by their common estimators:

$$\mathbb{E}\left(f(X)\right) \simeq \frac{1}{n}\sum_{i=1}^{n} f(X^{(i)}) \quad \text{Cov}(X) \simeq \frac{1}{n}\sum_{i=1}^{n}\left(X^{(i)} - \bar{X}\right)\left(X^{(i)} - \bar{X}\right)^T \text{ with } \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X^{(i)}$$

The problem (8.3) is basically a constrained multivariate minimization problem (with constraint $\|u\| = 1$). Despite its non-convexity in the general case, we can easily be convinced that it is locally convex. Therefore, having a smart initialization is paramount

to ensure reaching the minimum with common methods like Sequential Least Square Programming (SLSQP, [69]) or Constrained Optimization By Linear Approximation (COBYLA, [101]).

HFTU tries to reduce the variance of the input data. The direction gathering the maximum of variance is then a relevant initial guess for the normal vector u. Actually, this direction is given by the first principal component, which is the unit eigenvector corresponding to the greatest eigenvalues of the covariance matrix Σ. So it can efficiently be computed with the LOBPCG method [67] (Locally Optimal Block Preconditioned Conjugate Gradient) or merely with Power iterations for examples.

Algorithm 11 Computation of $\Phi(X)$

Require: X ($n \times d$ matrix)
Ensure: $\Phi(X)$
1: $\Sigma \leftarrow \mathrm{Cov}(X)$
2: $i \leftarrow \mathrm{argmax}_{k \in \{1,\dots d\}} \Sigma_{k,k}$ ▷ Canonical direction with the maximum of variance
3: $u^{(init)} \leftarrow \mathrm{LOBPCG}(\Sigma, e_i)$ ▷ Direction with the maximum of variance
4: $u_0^{(init)} \leftarrow -\mathbb{E}(X)^T u^{(init)}$ ▷ Selecting the hyperplane passing by $\mathbb{E}(X)$
5: $V_{(u,u_0)}(X) \leftarrow \mathrm{SLSQP}\left(V_\lambda(X), \lambda^{(init)} = \left(u^{(init)}, u_0^{(init)}\right)\right)$
6: $\phi(X) \leftarrow V_{(u,u_0)}(X) / \mathrm{Tr}\,\Sigma$
7: **return** $\phi(X)/\phi_d$

The algorithm 11 details the whole practical procedure to compute $\Phi(X)$. The first step is the covariance computation (line 1), needed both to initialize the optimizer and to compute the folding ratio $\phi(X)$ (line 6). The real optimization procedure (line 5) occurs after finding a smart starting point (lines 2 and 4). About the normal vector $u^{(init)}$ we have mentioned that we can use an eigenvector algorithm, however the latter needs also an initial guess. In our implementation, we merely use the canonical direction which gathers the maximum of variance (line 2). About the intercept $u_0^{(init)}$ we choose to start within the data, so we select the hyperplane passing by $\mathbb{E}(X)$.

Basically the complexity of the algorithm 11 is $O(n \cdot d + d^2)$: computing V_λ needs $O(n \cdot d)$ operations while SLSQP embeds additional $O(d^2)$ operations for matrix updates. However, the number of iterations of the optimizer is unknown so highlighting the real performance is rather hard (time performances are given in the next section).

Practical HFTU The theoretical HFTU can be unadapted in some limit cases. Let us imagine a sample $U^{(1)}, U^{(2)}, \dots U^{(n)}$ drawn uniformly within the $d-$ball. It may just as easily be considered unimodal or multimodal. Thus, it leads us to use significance levels to estimate the distance from the problematic case, namely the uniform distribution.

Retrieving the law of $Y_{d,n} = \Phi\left(U^{(1)}, U^{(2)}, \dots U^{(n)}\right)$ when $U^{(i)}$ are n i.i.d. random variables drawn uniformly within the $d-$ball seems very hard, that is why we rather perform Monte-Carlo simulations to estimate it. In our experiments, we have noticed

that $Y_{d,n}$ can be well modeled by a gaussian distribution with the following parameters:

$$\mu(n,d) = 1 - \frac{a_\mu \log d}{d\sqrt{n}} \quad \sigma(n,d) = \frac{a_\sigma}{d\sqrt{n}} \quad \text{with} \quad a_\mu \simeq 1.8 \text{ and } a_\sigma \simeq 0.9$$

Thus, we use in practice a very low quantile of $Y_{d,n}$ as the decision boundary, assuming that uniform distributions should be considered as unimodal. In the statistical hypothesis testing framework, it corresponds to the quantile related to the type I error α with null hypothesis H_0 : "X is uniformly distributed within a $d-$ball".

8.6 Experiments

In this section, we show the features of HFTU through different experiments. First, we compare the efficiency of FTU and HFTU, pointing out the weakness of the former test. Then, we detail the hyperplane provided by HFTU, using the SVM one as a baseline. For this purpose, we developed a `python3` library to compute HFTU [1]. and we also used the available `python3` implementation of FTU [2].

Comparison with FTU As HFTU gives another generalization than FTU does, we have to emphasize its advantages. In this paragraph, simple experiments on gaussian blobs are presented to check whether the unimodality/multimodality character is well detected. In each context (dimension, number of clusters), 500 random configurations are generated to evaluate FTU and HFTU. Every dataset gathers 2000 samples and the type I error α is set to 10^{-10} for HFTU (the $p-$value of the FTU is also set to 10^{-10}).

Case	#clusters	Test / Dimensions	2	3	5	10	15	20	30	50
Unimodal	1	FTU	100	100	100	100	100	100	100	100
		HFTU	100	100	100	100	100	100	100	100
Multimodal	5	FTU	83.4	94.8	100	100	100	100	100	100
		HFTU	99	100	99.8	99.4	99.4	99	98.6	98.6
Multimodal	10	FTU	62.6	69.4	82	100	100	100	100	100
		HFTU	94.6	99.6	99.8	99.8	99.4	99.4	99.6	98
Multimodal	15	FTU	57	54	59.2	87.6	100	100	100	100
		HFTU	91.8	99.8	99.8	100	100	99.2	99.2	99.2

Table 8.1: *Percentage of datasets rightly classified (unimodal or multimodal) according to the dimension and the number of clusters.*

All the results are given on table 8.1. First of all, both FTU and HFTU are unconditionally effective when the dataset is unimodal while difference arise in the multimodal case.

We can notice that the recall of HFTU is sometimes slightly lower that those of FTU. It mainly stems from the optimization procedure HFTU uses. In some cases, it does not

[1] available at `https://github.com/asiffer/hftu`

[2] available at `https://github.com/asiffer/python3-libfolding`

reach the absolute minima. This behaviour is likely to be a little amplified when the number of clusters increases because it also increases the number of local minima.

The most striking phenomenon is that FTU results collapse when the number of clusters increases, especially in low dimensions. In fact, it illustrates the main weakness of FTU described in section 8.1: the lack of robustness when facing configurations with some symmetries. When the number of clusters is high, symmetries are more likely to occur (particularly in low dimensions) and FTU can fail. HFTU is clearly more steady and robust when facing these distributions making it more reliable in the general case.

Time performance. Here we compare the time performances of HFTU with FTU according to the number of samples n and the dimension d. These tests have been performed on a single core Intel i5-8250U (with 16GB RAM). Two experiments have been made: first we set $d = 2$ and the average running time of FTU and HFTU are retrieved for several values of n (1000 runs for each n), second we set $n = 10000$ and we retrieve the timings for different values of d (1000 runs for each value of d).

Results are given on figures 8.4 and 8.3. Like FTU, we can see the linear complexity of HFTU with regards to the number of observations n. The dependency on the dimension is unfortunately not as clear. When the dimension increases we can observe the $O(d^2)$ complexity of HFTU (when n is constant) in comparison with FTU whose complexity with regards to d is still rather linear. Whereas these experiments show that FTU is faster than HFTU, we may also notice that HFTU can be used on several common contexts.

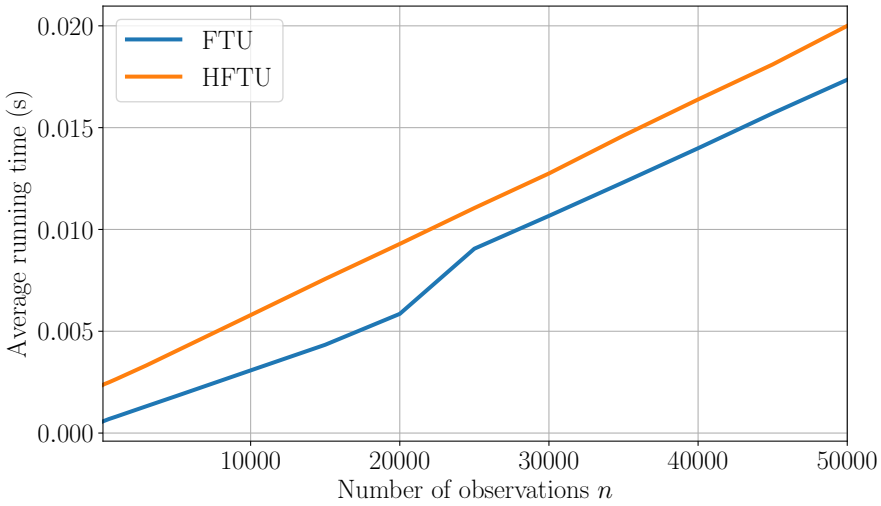


Figure 8.3: *Time performances of FTU and HTU according to the number of observations ($d = 2$)*

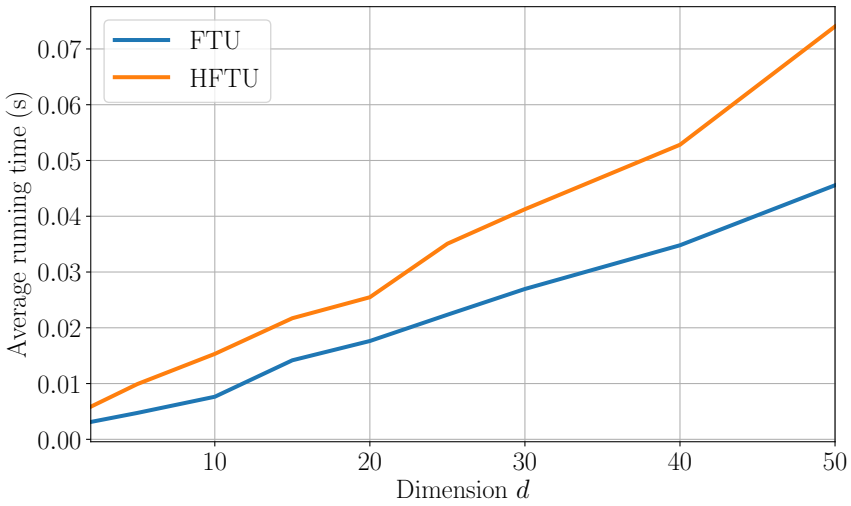


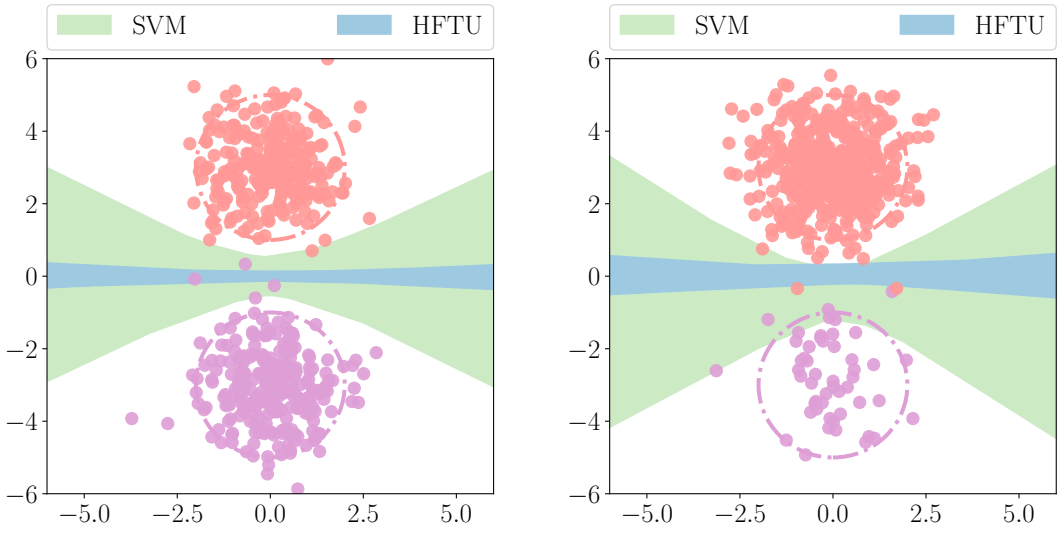
Figure 8.4: *Time performances of FTU and HTU according to the dimension ($n = 10000$)*

Robust hyperplane Finally we analyze the features of $\mathcal{H}^*$, the hyperplane provided by HFTU. We show it gives a robust macroscopic description of the distribution underlying the data. We merely use the hyperplane built by SVM [21], which is more local sensitive, as a baseline to present $\mathcal{H}^*$ features. In the two next experiments we analyze the positions of the provided hyperplanes computed on different samples drawn from the mixture $e\mathcal{N}(\mu_1, \sigma_1 I) + (1-e)\mathcal{N}(\mu_2, \sigma_2 I)$, with $\mu_1 = (0 \ \ -3)^T$ and $\mu_2 = (0 \ \ 3)^T$.

First, 2000 samples of 500 observations are generated in the balanced case ($e = 0.5$) and in an unbalanced case ($e = 0.1$) with $\sigma_1 = \sigma_2 = 1$. The figure 8.5 shows areas swept by each hyperplane.

Firstly, we can notice that $\mathcal{H}^*$ separates the two modes, so by solving the problem (8.3), HFTU actually builds an hyperplane with a relevant bisect capacity. Then, the areas swept by $\mathcal{H}^*$ in both cases are very thin, meaning that HFTU is robust against the sampling. As SVM works locally, its hyperplane is naturally more sensitive to the sampled observations. Moreover, we notice that the average position of $\mathcal{H}^*$ is the same between (a) and (b) while SVM hyperplane drifts towards the low density cluster. It shows that $\mathcal{H}^*$ is also robust to the cluster imbalance.

On the second experiment, we keep the balanced configuration but we tune the isotropic standard deviation σ_1 of the upper cluster (increasing the noise in the lower cluster). For each value of σ_1, 1500 samples of 500 observations are generated and the hyperplanes built by HFTU and SVM are retrieved. As we could not represent the effect on the hyperplanes as well as above, we rather choose to look at the angle of their normal vectors (with $e_1 = (1 \ 0)$) and their intercepts according to the standard deviation (see figures 8.6 and 8.7). Once again, $\mathcal{H}^*$ still separates the two modes (angle



(a) *Class balancing* $50\% - 50\%$ **(b)** *Class balancing* $90\% - 10\%$

Figure 8.5: *Comparison of the areas swept by HFTU and SVM hyperplanes in two balancing contexts (a single data sample is plotted to illustrate the clusters).*

$\simeq 90°$, intercept $\simeq 0$) and is very stable as both angle and intercept remain quite the same when σ_1 changes. So HFTU hyperplane is also robust against the noise.

8.7 Conclusion

HFTU is a new multivariate unimodality test. Compared to FTU, it has strong theoretical guarantees, eliminating the requirement of approximations. Such a property leads HFTU to solve the issues encountered by FTU. In addition to testing unimodality, HFTU computes an hyperplane $\mathcal{H}^*$ that provides a macroscopic description of the underlying distribution. $\mathcal{H}^*$ is particularly robust to sampling, data imbalance and noise. All these assets could make HFTU a relevant component of a clustering algorithm (sse the next chapter). These improvements are actually at the expense of the computational aspect since HFTU solves an optimization problem. However, we insist on the fact that FTU and HFTU can coexist since they have different features (table 8.2): FTU favors the simplicity and the speed while HFTU focuses on robustness and provides more information about data.

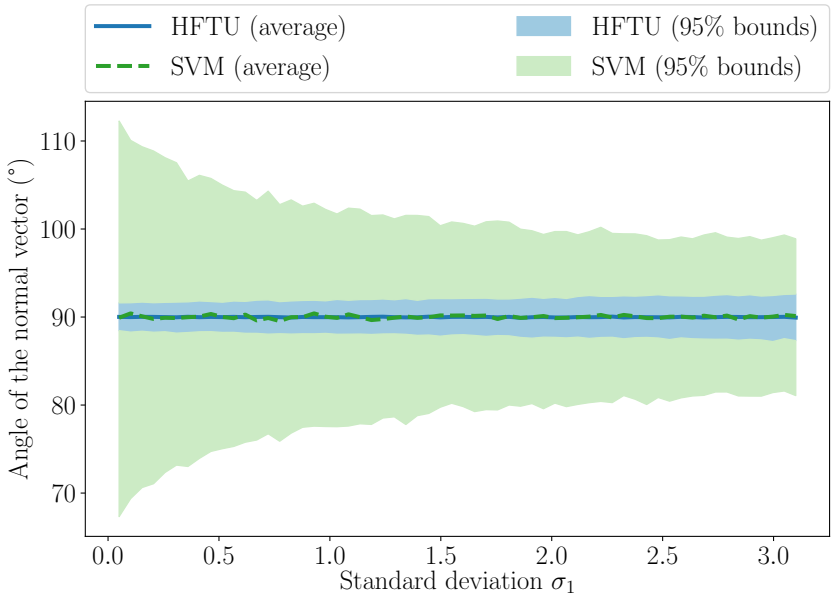


Figure 8.6: *Comparison of the HFTU and SVM hyperplanes normal vector angle when the standard deviation of one cluster increases. For each σ_1, the average behaviour and the 95% confidence interval are represented.*

Aspect	FTU	HFTU
Folding support	point s^*	hyperplane $\mathcal{H}^*$
Approximated support	yes	no
Computation of the support	analytical formula	optimization problem
Amount of information embedded in the support	low	high
Robust to symmetries	not in low dimensions	yes
Incremental computation	yes	no

Table 8.2: *Comparison of FTU/HFTU features*

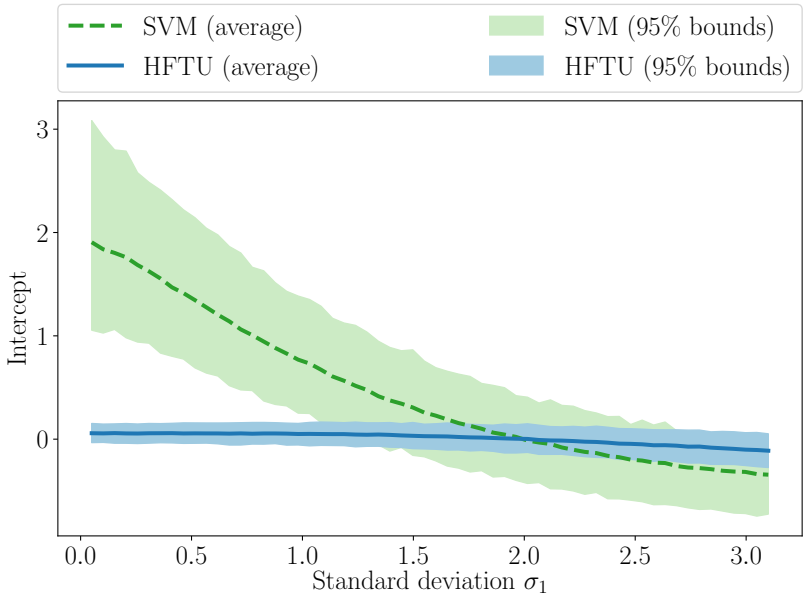


Figure 8.7: *Comparison of the HFTU and SVM hyperplanes intercept when the standard deviation of one cluster increases. For each σ_1, the average behaviour and the 95% confidence interval are represented.*

9

Clustering under unimodality assumption of the clusters

This last chapter deals with a direct application of unimodality testing: clustering. In fact, the definition of a "cluster" is subjective and every clustering algorithm states explicitly or implicitly what it considers as a cluster. From the statistician point of view, looking for clusters actually means finding high-density regions in the underlying data distribution. One common assumption is to look for gaussian-shaped clusters. Implicitly it fits the data distribution with gaussian mixtures. Unfortunately, such hypothesis is rather restrictive and is likely to produce bad clustering results once it is not verified.

More generally, a cluster can be claimed as valid once the underlying distribution is considered as unimodal. This principle is called the *unimodality assumption of the clusters* and tends to coincide with what we may empirically consider as a cluster. In particular, the aim of this section is to used HFTU to estimate the unimodality character of the clusters. We will first present another meaningful way to compute HFTU. This heuristic has several advantages that will be used to design an incremental clustering algorithm wrapping around $k-$means, named $\Phi-$means. The $\Phi-$means algorithm will be compared to other wrappers, showing that it is competitive to the state-of-art but

with weaker assumptions. The earliness of this research makes it even more promising through possible improvements we will detail in the end.

9.1 A $k-$means$-$like procedure to compute HFTU

In this section, we investigate another view of HFTU based on the law of total covariance. We will see that the problem solved by HFTU is addressed in a different way, leading to a $k-$means$-$like heuristic to perform it. This procedure will be used afterward for an iterative clustering algorithm.

Folding invariant First we need to reformulate the problem solved by HFTU (problem 8.3). For that purpose, let us suppose a binary random variable Z which divides the support of X in two parts. If X is only known through observations $X_1, \dots X_n$, we may see Z as binary classification of them (we have then two clusters). The covariance of X can be decomposed through the law of total covariance:

$$\mathrm{Cov}(X) = \underbrace{\mathbb{E}_Z\left(\mathrm{Cov}(X|Z)\right)}_{\text{Intra-cluster covariance}} + \underbrace{\mathrm{Cov}_Z\left(\mathbb{E}(X|Z)\right)}_{\text{Inter-cluster covariance}} .$$

And we can apply the trace operator (which is linear) as follows

$$\mathrm{Tr}\,\mathrm{Cov}(X) = \underbrace{\mathbb{E}_Z\left(\mathrm{Tr}\,\mathrm{Cov}(X|Z)\right)}_{\text{Intra-cluster overall variance}} + \underbrace{\mathrm{Tr}\,\mathrm{Cov}_Z\left(\mathbb{E}(X|Z)\right)}_{\text{Inter-cluster overall variance}} . \tag{9.1}$$

The first term is the average of the clusters variance (sum of the variances in every direction). The second is term is not as obvious, but it is related to the gap between the two clusters.

Given an hyperplane $\mathcal{H}_\lambda$, we can take $Z = \mathrm{sign}\left(h_\lambda(X)\right)$ (we recall that h_λ is defined in def. 8.1). Thus $Z = 1$ when X is above $\mathcal{H}_\lambda$ and $Z = -1$ when X is below $\mathcal{H}_\lambda$. The first term in (9.1) is the average between the "variance above $\mathcal{H}_\lambda$" and the "variance below $\mathcal{H}_\lambda$". Intuitively it does not change through folding about $\mathcal{H}_\lambda$, because the covariance does not depend of the location: data below are just mapped above and their overall variance remains the same. The proposition 9.1 shows it formally.

PROPOSITION 9.1 *Let X be random variable such that $\mathrm{Cov}(X)$ exists. Let $\lambda = (u, u_0) \in \mathcal{S}^d \times \mathbb{R}$, we note $Z = \mathrm{sign}\left(h_\lambda(X)\right)$. We have*

$$\mathbb{E}_Z\left(\mathrm{Tr}\,\mathrm{Cov}(\mathrm{Fold}_\lambda(X)|Z)\right) = \mathbb{E}_Z\left(\mathrm{Tr}\,\mathrm{Cov}(X|Z)\right).$$

Proof.

$$\begin{aligned}\mathbb{E}_Z\left(\mathrm{Cov}\left(\mathrm{Fold}_\lambda(X)\,|\,Z\right)\right) = {} & \mathrm{Cov}\left(\mathrm{Fold}_\lambda(X)\,|\,Z=1\right)\mathbb{P}(Z=1) \\ & + \mathrm{Cov}\left(\mathrm{Fold}_\lambda(X)\,|\,Z=-1\right)\mathbb{P}(Z=-1)\end{aligned}$$

But every term can be simplified (we recall that $\text{Fold}_\lambda(X) = X - (h_\lambda(X) - |h_\lambda(X)|)\, u$)

$$\begin{aligned}\text{Cov}\,(\,\text{Fold}_\lambda(X)\,|\,Z=1) &= \text{Cov}\,(\,X-(h_\lambda(X)-|h_\lambda(X)|)\,u\,|\,h_\lambda(X)\geq 0)\\ &= \text{Cov}\,(\,X\,|\,h_\lambda(X)\geq 0)\end{aligned}$$

$$\begin{aligned}\text{Cov}\,(\,\text{Fold}_\lambda(X)\,|\,Z=-1) &= \text{Cov}\,(\,X-(h_\lambda(X)-|h_\lambda(X)|)\,u\,|\,h_\lambda(X)<0)\\ &= \text{Cov}\,(\,X-2h_\lambda(X)u\,|\,h_\lambda(X)<0)\\ &= \text{Cov}\left(X-2u^TXu\,\middle|\,h_\lambda(X)<0\right)\\ &= \text{Cov}\left(\left(\mathbf{I}_d-2uu^T\right)X\,\middle|\,h_\lambda(X)<0\right)\\ &= A\,\text{Cov}\,(\,X\,|\,h_\lambda(X)<0)\,A^T = A\,\text{Cov}\,(\,X\,|\,h_\lambda(X)<0)\,A\end{aligned}$$

with $A = \mathbf{I}_d - 2uu^T$ (symmetrical matrix). We also have

$$\text{Tr}\,[A\,\text{Cov}\,(\,X\,|\,h_\lambda(X)<0)\,A] = \text{Tr}\left[A^2\,\text{Cov}\,(\,X\,|\,h_\lambda(X)<0)\right] = \text{Tr}\,\text{Cov}\,(\,X\,|\,h_\lambda(X)<0)$$

because $A^2 = \left(\mathbf{I}_d-2uu^T\right)\left(\mathbf{I}_d-2uu^T\right) = \mathbf{I}_d - 4uu^T + 4uu^Tuu^T = \mathbf{I}_d$, as $u^Tu = \|u\|^2 = 1$. Eventually, with the linearity of the trace operator, we get

$$\begin{aligned}\text{Tr}\,[\mathbb{E}_Z\,(\text{Cov}(\text{Fold}_\lambda(X)|Z))] &= \text{Tr}\,[\text{Cov}\,(\,X\,|\,h_\lambda(X)\geq 0)]\,\mathbb{P}(Z=1)\\ &\quad + \text{Tr}\,[\text{Cov}\,(\,X\,|\,h_\lambda(X)<0)]\,\mathbb{P}(Z=-1)\\ &= \text{Tr}\,[\mathbb{E}_Z\,(\text{Cov}(X|Z))]\,,\end{aligned}$$

and, by inverting the two linear operators on both sides, we get the desired result. □

Alternative problem This folding invariant means that optimizing the folding stage is equivalent to minimizing the second term in (9.1). So HFTU solves the following minimization problem.

PROBLEM 9.1 *Let X be a twice integrable random variable with covariance matrix Σ. HFTU solves the following optimization problem:*

$$\min_{\lambda=(u,u_0)\in\mathcal{S}^d\times\mathbb{R}} \text{Tr}\,\text{Cov}\,[\mathbb{E}\,(\,\text{Fold}_\lambda(X)\,|\,\text{sign}\,h_\lambda(X))] \tag{9.2}$$

To understand what the formula (9.2) really means, we note W_λ the random variable of $\mathbb{R}^d$ defined by $W_\lambda = \mathbb{E}(\text{Fold}_\lambda(X)|\,\text{sign}\,(h_\lambda(X))$. W_λ can take only two values (in $\mathbb{R}^d$) according to the sign of $h_\lambda(X)$. Let us give its distribution:

W_λ	$a_\lambda = \mathbb{E}(X\,\|\,h_\lambda(X)\geq 0)$	$b_\lambda = \mathbb{E}(X-2h_\lambda(X)u\,\|\,h_\lambda(X)<0)$
$\mathbb{P}(W_\lambda)$	$\mathbb{P}(h_\lambda(X)\geq 0)$	$\mathbb{P}(h_\lambda(X)<0)$

The first possible value a_λ is the center of the points above $\mathcal{H}_\lambda$ while the second b_λ is the center of the points below $\mathcal{H}_\lambda$, folded about $\mathcal{H}_\lambda$. Minimizing $\operatorname{Tr}\operatorname{Cov} W_\lambda$ boils down to fold the lower center onto the upper one. We have to notice that neither $\mathbb{P}(h_\lambda(X) \geq 0)$ nor $\mathbb{P}(h_\lambda(X) < 0)$ can be equal to zero, otherwise the conditional expected values are not defined. The proposition 9.2 particularly shows that $\operatorname{Tr}\operatorname{Cov} W_\lambda$ is minimal when $a_\lambda = b_\lambda$. The figure 9.1 illustrates this optimization made by HFTU.

PROPOSITION 9.2 *Let X be random variable such that $\operatorname{Cov}(X)$ exists. Let $\lambda = (u, u_0) \in \mathcal{S}^d \times \mathbb{R}$ such that $q_\lambda = \mathbb{P}(h_\lambda(X) \geq 0) \in (0, 1)$. We have*

$$\operatorname{Tr}\operatorname{Cov} W_\lambda = q_\lambda(1 - q_\lambda) \left\| a_\lambda - b_\lambda \right\|^2 \tag{9.3}$$

Proof. We only need to develop the covariance:

$$\begin{aligned}
\operatorname{Cov} W_\lambda = \operatorname{Cov}(W_\lambda, W_\lambda) &= \mathbb{E}(W_\lambda W_\lambda^T) - \mathbb{E}(W_\lambda)\,\mathbb{E}(W_\lambda)^T \\
&= q_\lambda a_\lambda a_\lambda^T + (1-\alpha) b_\lambda b_\lambda^T \\
&\quad - (q_\lambda a_\lambda + (1-q_\lambda) b_\lambda)\,(q_\lambda a_\lambda + (1-q_\lambda) b_\lambda)^T \\
&= q_\lambda(1-q_\lambda) a_\lambda a_\lambda^T + q_\lambda(1-q_\lambda) b_\lambda b_\lambda^T \\
&\quad - q_\lambda(1-q_\lambda)(a_\lambda b_\lambda^T + b_\lambda a_\lambda^T) \\
&= q_\lambda(1-q_\lambda)\left(a_\lambda a_\lambda^T + b_\lambda b_\lambda^T - b_\lambda a_\lambda^T - a_\lambda b_\lambda^T\right).
\end{aligned}$$

So, by taking the trace we get

$$\operatorname{Tr}\operatorname{Cov} W_\lambda = q_\lambda(1-q_\lambda)\left(\|a_\lambda\|^2 + \|b_\lambda\|^2 - 2 a_\lambda^T b_\lambda\right) = q_\lambda(1-q_\lambda)\left\|a_\lambda - b_\lambda\right\|^2.$$

□

Heuristic computation of $\mathcal{H}^*$ We may wonder whether the new view given by the problem 9.1 and the proposition 9.2 is easier to tackle. Actually, the equation (9.3) is not a simple distance optimization since every term depends on λ. For instance estimating the behavior of a_λ (or b_λ) with λ is very hard in the general case. However, it leads to a simple heuristic to find $\mathcal{H}^*$, close to the $k-$means algorithm. At a given iteration let us assume that we have an hyperplane $\mathcal{H}_\lambda$. We can compute the centers a_λ, b_λ and finally update $\mathcal{H}_\lambda$ to make a_λ and b_λ merge. This step can be performed until convergence, in the same manner as $k-$means.

The algorithm 12 details this iterative way to compute $\mathcal{H}^*$. First of all, we need to initialize the hyperplane (line 1). For instance we can use the same initialization as previously, i.e. the hyperplane orthogonal to the direction of maximum variance and passing through the mean of the dataset. The value of the target function (equation 9.3) and the centers of the two clusters are computed on line 2. They are computed with the algorithm 13 we detail later. Then at every iteration, the hyperplane is updated so has to merge both centers (lines 5 and 15). The new normal vector is computed to be collinear

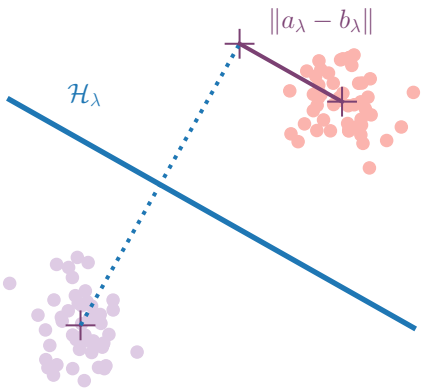


Figure 9.1: *Alternative problem solved by HFTU: $\mathcal{H}^*$ is searched so as to minimize the distance between the upper center a_λ and the folded lower center b_λ*

with the vector made by the two centers (oriented towards the upper center, line 5). The new intercept is computed to make $\mathcal{H}$ pass through the middle of the two centers (line 15). Thus, they are symmetrical about $\mathcal{H}$. Finally, we add two convergence parameters N and ε. N is the maximum number of iterations while ε is the convergence criterion: the algorithm is stopped when the value of the target function is stable enough.

The function to minimize is given by the equation (9.3) and the way to computed it is detailed by the algorithm 13. This latter procedure also returns the centers of each cluster ("above" and "below"). Briefly, the algorithm splits the points according to their position about the hyperplane and computes the center of both groups. It also computes the ratio of points above $\mathcal{H}$ which estimates p_λ. These quantities are notably used to compute $\operatorname{Tr}\operatorname{Cov} W_\lambda$ (line 17).

The asymptotic complexity of one iteration of the algorithm mainly depends on the TARGET function so it merely needs $O(n \cdot d)$ operations which scales better than the initial algorithm computing $\mathcal{H}^*$ (algorithm 11 with $O(n \cdot d + d^2)$ operations in a single SLSQP iteration). These results will be highlighted in the next section. This heuristic is also far simpler and better controls the number of iterations while the algorithm 11 relies on a black-box constrained optimization algorithm. Finally, it may also output some side information (like the position of the above and below centers) which will be paramount to build an incremental clustering algorithm.

REMARK 9.1 *As in $k-$means, initialization is a crucial step, especially to make it converge to the best hyperplane (minimizing Φ). So, the algorithm 12 can also be run with a random initialization. But, we must ensure that a_λ and b_λ exist, namely we must*

Algorithm 12 Heuristic computation of $\mathcal{H}^*$

Require: X ($n \times d$ matrix), N, ε
Ensure: $\mathcal{H}^*$
$u, u_0 \leftarrow \text{INIT}(X)$ ▷ get an initial hyperplane
$W, A, B \leftarrow \text{TARGET}(X, u, u_0)$ ▷ compute the objective function
$i \leftarrow 0$
while $i < N$ **do**
  $v \leftarrow (A - B)/\|A - B\|$ ▷ update the hyperplane
  $v_0 \leftarrow -(1/2)\, v^T (A + B)$
  $w, a, b \leftarrow \text{TARGET}(X, v, v_0)$ ▷ re-compute the objective function
  if $|(W - w)/W| < \varepsilon$ **then** ▷ check convergence
    if $w < W$ **then**
      return v, v_0
    else
      return u, u_0
    end if
  end if
  $u_0 \leftarrow v_0$
  $u \leftarrow v$
  $W, A, B \leftarrow w, a, b$
  $i \leftarrow i + 1$
end while
return u, u_0

have data on both sides of $\mathcal{H}$. In practice, we coerce the initial hyperplane into passing through $\bar{X}$ after a random selection of the normal vector.

9.2 Experimental comparison of the HFTU computations

In this paragraph, we compare the HFTU algorithms, called "legacy" (algorithm 11) and "heuristic" (algorithm 12). For this latter algorithm we may inspect two possible initializations: "variance" (choose the hyperplane orthogonal to the maximum variance direction) or "random" (choose a random hyperplane). In the random initialization case, several runs can be performed so as to retrieve the best one. We have developed an `python3` package, called `hftu`[1], which implements both methods and both initializations to compute HFTU. For the heuristic, we use $N = 30$ and $\varepsilon = 10^{-5}$, while the legacy algorithm use the default parameter values of the SLSQP implementation of the `scipy` library.

[1] `https://github.com/asiffer/hftu`

Algorithm 13 Target function (computation of $\operatorname{Tr}\operatorname{Cov} W_\lambda$)

procedure TARGET(X, u, u_0)
  $A \leftarrow 0, B \leftarrow 0$
  $n \leftarrow 0, q \leftarrow 0$
  for $x \in X$ **do**
    $n \leftarrow n + 1$ ▷ increment the number of observations
    $h \leftarrow u^T x + u_0$ ▷ check the side of x about $\mathcal{H}$
    if $h \geq 0$ **then**
      $A \leftarrow A + x$
      $q \leftarrow q + 1$ ▷ increment the number of observations above $\mathcal{H}$
    else
      $B \leftarrow B + x$
    end if
  end for
  $A \leftarrow A/q$ ▷ center of points above $\mathcal{H}$
  $B \leftarrow B/(n - q)$ ▷ center of points below $\mathcal{H}$
  $p \leftarrow q/n$ ▷ ratio of points above $\mathcal{H}$
  $W \leftarrow p(1 - p) \times \|A - B\|^2$ ▷ objective function
  return W, A, B
end procedure

Efficiency First we compare the performances of the two algorithms on a hard context: low dimension, many clusters. In this context, we have seen that FTU totally collapses while HFTU (legacy version) looks more robust (see the previous results on table 8.1). However, this situation seems to be the worst for HFTU. For that purpose, we generate 5000 datasets with 2000 points in dimension 2 gathered in 15 clusters. For each dataset, the folding statistics Φ is computed in several ways (legacy or heuristic) and compared to the decision bound at level 10^{-10} (the same as in the experiment 8.6). Thus we can retrieve the ratio of configurations which are rightly classified as multimodal. Results are presented on figure 9.2. We can see the performances of the legacy and heuristic computations. For the heuristic computation we can also check the differences according to the initialization ("variance" or "random").

First we notice that the heuristic with variance initialization does not provide better results than the legacy computations. However, the random initialization makes the heuristic better once 5 runs are performed. Furthermore, we may observe that increasing the number of runs does not provide significant improvements. How to interpret these results? We may recall that the legacy computation of HFTU uses the "variance" initialization. So, even if the constrained optimizing procedure allows to find a good solution, this initialization may lead the algorithm not to the optimum while a random initialization explores intuitively more the solution space. Besides, few random runs are required meaning that the heuristic does converge to good solutions.

We may wonder whether a random initialization is able to improve the legacy

algorithm. In our experiments we have observed that the results are not significantly better with this initialization while the computation time is far more impacted.

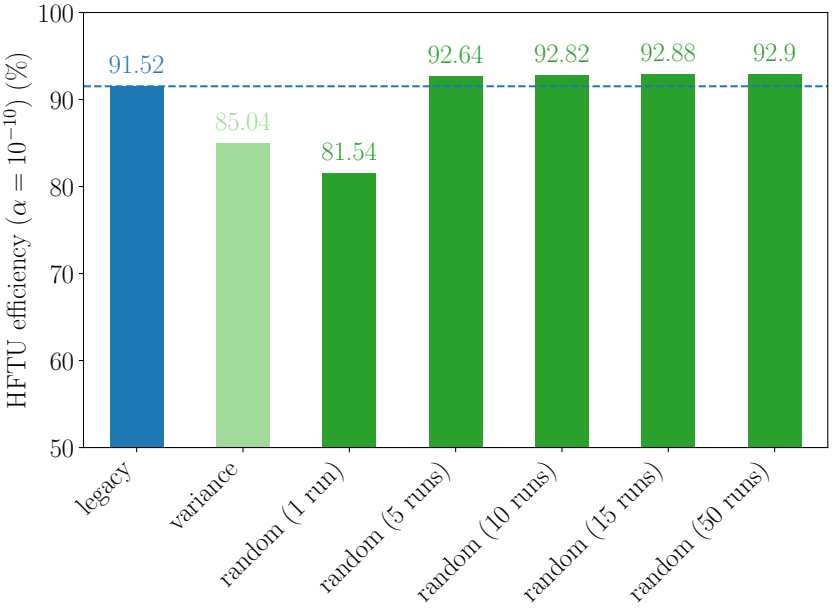


Figure 9.2: *Performances of HFTU according to the algorithm on multimodal datasets. The legacy computation is in blue (first bar) while green color is used to denote the heuristic (other bars), with different initializations*

Time performances Finally, we may check the running time of both algorithms. All the experiments use a single core Intel Xeon E5-2695 v4 @ 2.10GHz.

To compare the dependency in the number of observations n, we generate 2000 random datasets in dimension 3 for different values of n. We retrieve then the average time to compute HFTU according to the algorithm. Results are presented on figure 9.3. First we notice that the complexity of both algorithms is linear with respect to n. However, the slope is lower for the legacy version. The table 9.1 details more the gaps (multiplicative factors) between the heuristic and the legacy algorithm. For instance, 5 runs of the heuristic depend about 3 times more on n than the legacy algorithm.

Heuristic	1 run	5 runs	15 runs
Comparison with legacy	$\times 1.3$	$\times 3.0$	$\times 7.5$

Table 9.1: *Linear dependency in the number of observations n of the heuristic in comparison with the legacy algorithm*

About the dependency in the dimension d, we generate 2000 random datasets of 8000 points for different values of d. The average time to perform HFTU is also computed

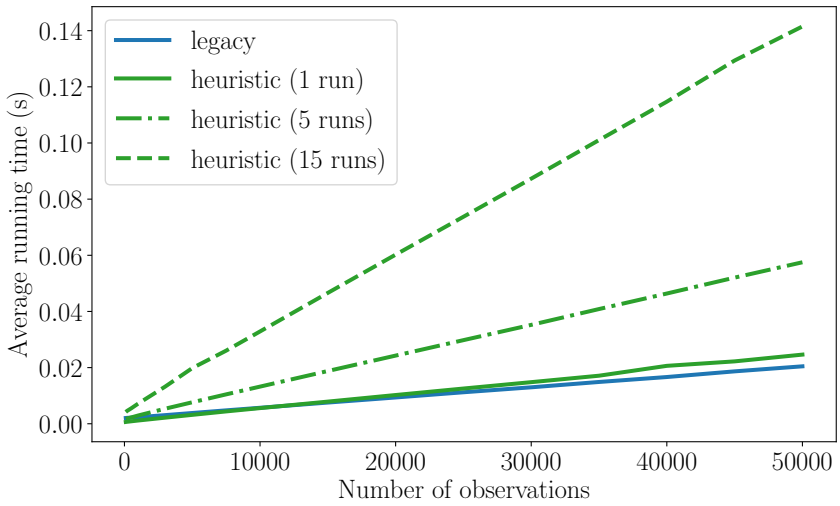


Figure 9.3: *Time performances according to the number of observations (dimension 3)*

according to the algorithm used. The figure 9.4 shows the results. We may also observe the linear complexity of the heuristic with respect to the dimension d, therefore it scales better than the legacy algorithm with $O(d^2)$ complexity.

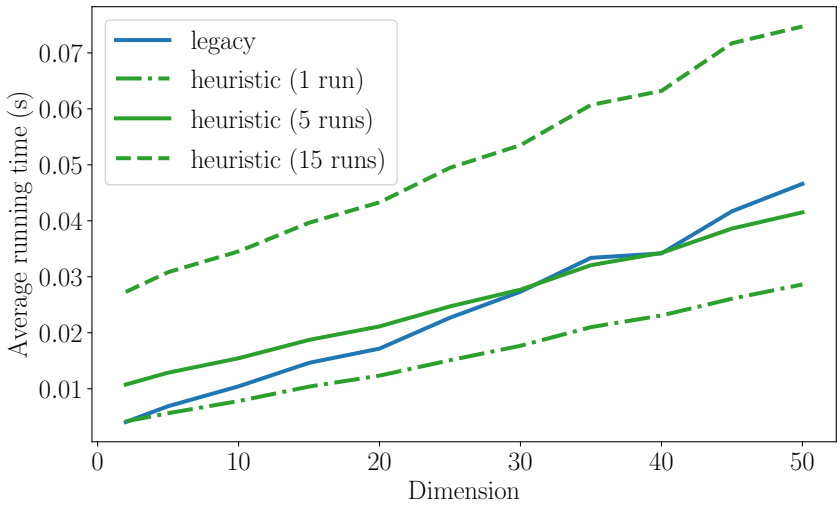


Figure 9.4: *Time performances according to the dimension (8000 observations)*

9.3 The Φ−means algorithm

Undoubtedly $k-$means is a widely used clustering algorithm. Its main drawback is that it requires to know beforehand the number k of clusters. Numerous approaches propose wrapping strategies around the $k-$means algorithm, that automatically find a value for k.

In these techniques a scoring process is needed to validate the clusters. Approaches like $X-$means [98], $G-$means [50] or $PG-$means [32] use statistical tests but they rely on the assumption that the clusters are Gaussian, limiting their generality. In a non-Gaussian setting, these approaches have a tendency to overcluster. The *dip*−means algorithm [62] has attempted to solve this problem by relying on a weaker assumption: a cluster is valid when it is unimodal. For this purpose they use the dip-test of unimodality [53] (described in 5.3). Nevertheless, we note that this test is designed for univariate data, meaning that the *dip*−means algorithm has to rely on numerous pairwise comparisons to handle multidimensional data. This severly hampers the scalability of the approach.

To overcome all these limitations, we propose Φ−means, a new wrapper that also relies on the unimodality assumption of the clusters but based on HFTU. HFTU determines if a cluster is unimodal or not without any distribution assumption, but unlike the dip-test, HFTU is designed to handle multidimensional data, and has a linear time complexity in the number of samples. Φ−means inherit from these properties.

$k-$means wrappers Several works have already been proposed to find the number of clusters. Most of the approaches wrap around a common clustering algorithm like $k-$means (or the more general Expectation-Maximization algorithm). The guiding principle of these wrappers is to start with one cluster ($k = 1$) and to increase the number of clusters at each step until they are all "valid" according to a given criterion.

At the highest point of view, two strategies exist: the clustering can be enriched either globally, on the whole model, or locally, through each cluster. Globally, a raw $k-$means is run with $k = 1, k = 2 \ldots$ until all the output clusters respect a given criterion or a best score is reached. A common example is the *elbow* method which finds a "break" in the plot (k, inertia). On the other hand, local algorithms try to split recursively the clusters keeping in memory the previous steps. They may also refer to *incremental* algorithms. Among well known wrappers, $PG-$means [32] uses a global strategy while $X-$means [98], $G-$means [50], Bayesian $k-$means [71] and *dip*−means [62] work locally.

Without wrapping around $k-$means, other techniques finding the number of clusters exist like the gap statistic [124], the average silhouette [104, 79], the minimum distortion [121] or stability-based model validation [74]. These methods could have a link with global approaches but they are not closely related to our proposed method.

At each iteration, local algorithms have to decide whether to split clusters or not. For this purpose, they all use their own *scoring* method which really shapes the core of the algorithm. As we will explain, the main properties stem from it.

$X-$means uses the bayesian information criterion (BIC) to score the clusters. However, this choice implies two issues. First, the BIC is relative to a model as it notably requires the computation of the likelihood. The authors of [98] have then considered spherical gaussian distributions, which is rather restrictive. Second, the decision to keep or split a cluster cannot be taken only with a BIC: the single cluster model must be compared with the two clusters model (the model which has the lowest BIC is kept). So the method does not require any parameter but evaluating the two cluster model is only possible if the split has already been done.

The $G-$means scoring method suffers from the same problems. In particular, $G-$means uses a Anderson-Darling test [4] to decide whether a cluster is sampled from a Gaussian or not (gaussian assumption of the clusters). Furthermore, this test can be used only on univariate distributions so data have to be projected to one dimension (the projection of gaussian vectors are also gaussian). The choice of the projecting line is then paramount: in [50], the authors perform a split, retrieve the two new centers and take the line passing through them. This choice is quite smart to evaluate the single gaussian cluster model but it also requires a split.

dip$-$means is the first $k-$means wrapper not relying on gaussian assumption. Unlike $X-$means or $G-$means, the approach is based on the *unimodality assumption of the clusters*. In this context, data only needs to be drawn from a unimodal distribution to shape a valid cluster, which is far less restrictive than previous approaches. The scoring method in [62] is based on the aforementioned dip test of unimodality [53]. This test is very powerful but does not tackle multivariate distributions. To circumvent this issue, the authors consider the distribution of the pairwise distances within every cluster, building then the *dip-dist criterion*. For every point of a cluster x_i, the dip-test is run on the distances $\{\|x_i - x_j\|, j \neq i\}$. After that, a voting scheme allows the decision: If the proportion of x_i for which the dip-test fails is greater than a given threshold, the cluster is split. This technique eases the extension to kernel spaces, making possible the clustering of some complex structured clusters. However, its main drawback is the complexity: computing all the pairwise distances severely hinders scalability. In our point of view, it goes against the $k-$means philosophy which linearly depends on the number of observations.

Design of $\Phi-$*means* Here we detail $\Phi-$means, a new $k-$means wrapper based on the unimodality assumption of the cluster. It uses HFTU to both estimate the unimodality character and perform a possible split. Indeed, the key asset of HFTU is the hyperplane $\mathcal{H}^*$ it builds in addition to the test. $\mathcal{H}^*$ has a relevant bisection capacity that we use to incrementally split clusters. It notably avoids additional splitting procedure. In a word, HFTU embeds all the desired materials to build a straightforward clustering algorithm.

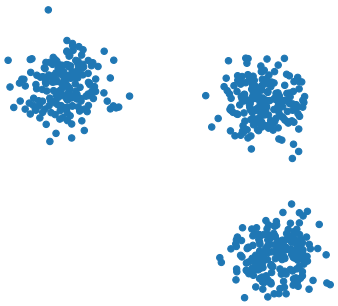
First let us comment the figure 9.5, which step-by-step details a run of the $\Phi-$means algorithm on a 3-clusters dataset. The guiding principle is the following: clusters are splitted until they are all unimodal. Initially, we consider a single cluster: the whole dataset (9.5a). HFTU is performed on this cluster (9.5b) and two key elements are retrieved: the unimodality character (which stems from the computed folding statistics

Φ) and the centers of both hyperplane sides (center of data "above", noted A, and center of data "below", noted B). The cluster being declared as multimodal, $k-$means is run with $k = 2$ starting from the two centers A, B (9.5c). Then the unimodality of the two new clusters is checked (9.5d). One cluster (red) is considered as unimodal while the other one (purple) should be split. So $k-$means is run with $k = 3$ and initial centers given by the center of the unimodal cluster (already known), and the two new centers output by HFTU (9.5e). Finally the unimodality of the three clusters output by $k-$means is checked and the algorithm stops (9.5f).

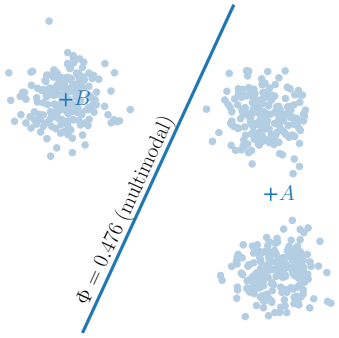
Algorithm Now we present the algorithm in a more practical way (see algorithm 14). On line 1, the algorithm starts with one cluster (the whole dataset) and one center (the mean of the data). A variable n is used to store the number of clusters before running HFTU. It allows to check whether the number of clusters has increased after HFTU (line 3). While new clusters appear, we run $k-$means on the whole dataset based on the current set of centers we have (line 5). This step is very fast because the greatest part of the starting centers will actually be very close to the solution (few iterations within $k-$means). Then $\Phi-$means scores every cluster with HFTU (line 8), and split them if necessary (line 12). "Splitting" merely means that the two new centers will be used for the next $k-$means. We may argue that if a cluster is unimodal then there is no need to re-compute HFTU, nonetheless the next $k-$means runs are also able to reshape it. $\Phi-$means naturally stops when all the clusters have been reported as unimodal, meaning that no new centers have been added.

Algorithm 14 $\Phi-$means

Require: X, α
Ensure: $Clusters, Centers$
1: $Centers \leftarrow \{\bar{X}\}, Clusters \leftarrow \{X\}$ ▷ set of centers/clusters
2: $n \leftarrow 0$
3: **while** $|Centers| \neq n$ **do** ▷ check if the number of centers increases
4: $n \leftarrow |Centers|$
5: $Clusters, Centers \leftarrow \text{KMEANS}(X, Centers)$
6: $M \leftarrow \{\}$ ▷ temporary set of centers
7: **for** m, Z **in** $Centers, Cluster$ **do**
8: $unimodal, a, b \leftarrow \text{HFTU}(Z, \alpha)$ ▷ score
9: **if** unimodal **then**
10: $M \leftarrow M \cup \{m\}$ ▷ do not split
11: **else**
12: $M \leftarrow M \cup \{a, b\}$ ▷ split according to $\mathcal{H}^*$
13: **end if**
14: **end for**
15: $Centers \leftarrow M$ ▷ set of centers that will be used for next k-means
16: **end while**

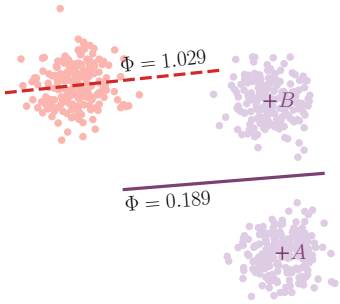


(a) *Initial dataset (a single cluster is assumed)*



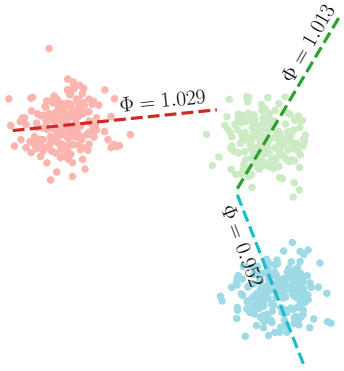
(b) *Run HFTU on these data and check if the cluster is unimodal or not (in the multimodal case, the "above" and "below" centers are kept)*

(c) *Run $k-$means assuming $k = 2$ and initialized with the two previous centers*



(d) *Run HFTU on each cluster, and check their unimodality. Here only one cluster must be split*

(e) *Run $k-$means assuming $k = 3$ and initialized with the new centers*



(f) *Check unimodality. Stop when all clusters are unimodal*

Figure 9.5: *$\Phi-$means algorithm. Computed hyperplanes are represented by either solid lines when it leads to split a cluster or dashed lines when the cluster is considered as unimodal*

We can also give some elements about its complexity. Let us consider a dataset of n $d-$dimensional observations gathered in k clusters. To well separate the clusters, $k-1$ splits must be made so $O(k)$ iterations of the main loop are required. During one iteration, $k-$means is run and HFTU is performed on every clusters. On the one hand, even if $k-$means is well initialized, it has a $O(n \cdot k \cdot d)$ complexity. One the other hand, the test is linear in the number of observations, so the batch of tests will cost merely $O(n \cdot d)$ if we consider the heuristic computation. Gathering all these stages, the overall complexity is $O(n \cdot k \cdot d)$. We may notice that this is not clear whether $k-$means or HFTU is the limiting factor, however the experiments presented in the next chapter will show that $k-$means is not much used (few iterations) making then HFTU rather predominant.

9.4 Experiments

In this section we experiment the properties of $\Phi-$means. In particular, we analyze the influence of its single parameter α and we compare the algorithm with other $k-$means wrappers. We will see that $\Phi-$means is clearly competitive with the state-of-the-art but with weaker assumptions. In addition, $\Phi-$means shows many more assets than its closest challenger, namely *dip*$-$means.

For these tests, we have developed a `python3` library called `wtvmeans` which implements $X-$means, $G-$means, *dip*$-$means, $\Phi-$means and the Elbow method. It is available on Github[2] and downloadable through the `pip` package manager.

Influence of α The $\Phi-$means algorithm needs a single parameter: the type I error α of HFTU. This parameter tunes the unimodal/multimodal boundary. When α is very low (close to zero), HFTU will be likely to consider clusters as unimodal. It means that $\Phi-$means will less split the cluster when α decreases. Formally, α is the probability (the risk) to claim "multimodal" while the distribution is unimodal. Intuitively, if this probability is low, HFTU tends to declare every distribution as unimodal to avoid being mistaken is case of the distribution is really unimodal. On the figure 9.6, we illustrate the behavior of $\Phi-$means for different values of α (from 10^{-10} to 10^{-60}) on different toy datasets.

First if all, we observe the main impact of α: $\Phi-$means stop splitting when α is very low (from 10^{-50} notably). For "high" values of α, $\Phi-$means is likely to over-split. On these examples we can see that $\Phi-$means is not unconditionally successful but we should recall that HFTU is less efficient in low dimension. It might explain why there is visibly not an obvious value for α. From this experiment, setting α between 10^{-40} and 10^{-30} seems reasonable.

Toy comparison Naturally, we want to compare $\Phi-$means with other $k-$means wrappers. We have to recall that these clustering algorithms cannot provide a better

[2] `https://github.com/asiffer/wtvmeans`

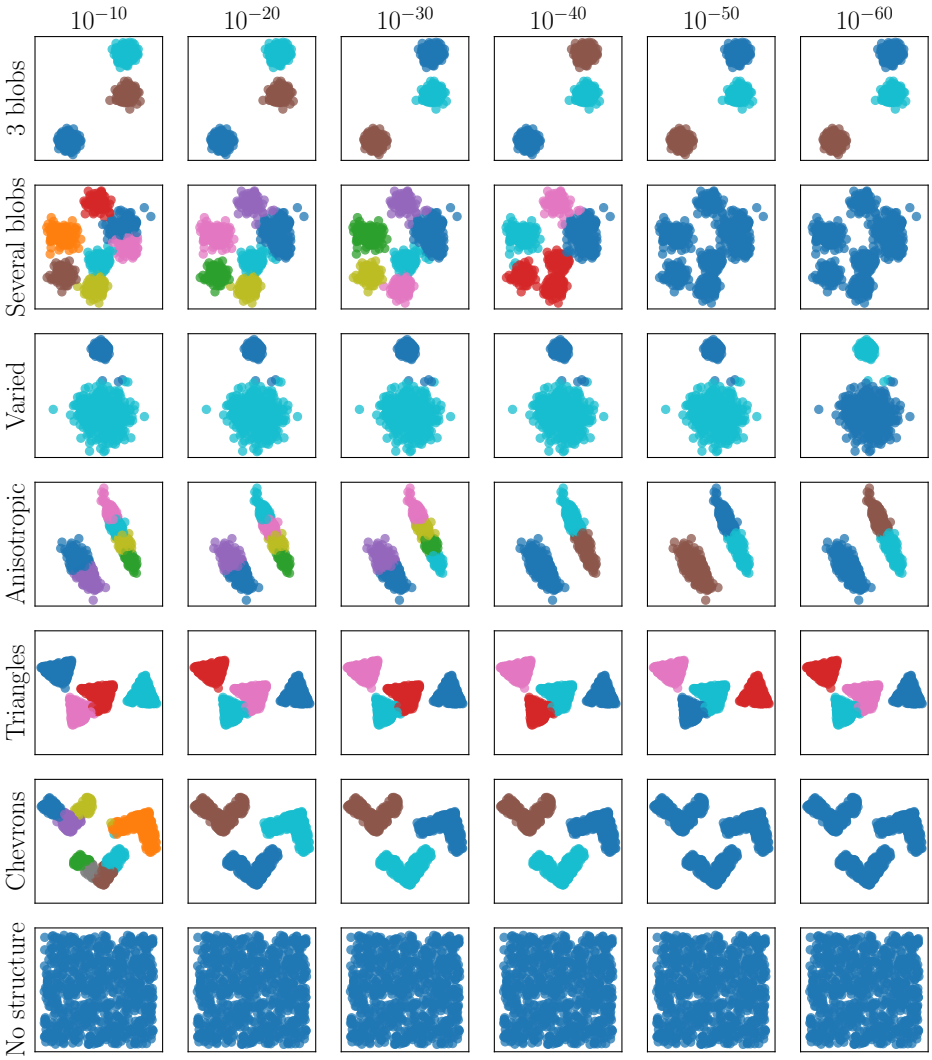


Figure 9.6: $\Phi-$*means: influence of the parameter* α *on different 2D toy datasets*

result than $k-$means because they use it to perform the clustering. However, they aim to find the *best* k.

As a baseline, we will use $k-$means with the "elbow" method. This is rather a visual process which finds a break in the plot $(k,\ \mathrm{SSE}_k)$ where SSE_k is the sum of squared errors (for k clusters) also known as *inertia*. To automatically find the best k we have chosen to use the following criterion:

$$k^* = \underset{k\in\{2,\dots k_{\max}\}}{\operatorname{argmax}} \frac{\mathrm{SSE}_{k-1} - \mathrm{SSE}_k}{\mathrm{SSE}_k - \mathrm{SSE}_{k+1}},$$

with $k_{\max} = 15$. We have to notice that the criterion looks for at least two clusters. In addition to Elbow, we compare $\Phi-$means with $G-$means, $X-$means and *dip*$-$means.

First we analyze the behavior of all the algorithms on toy datasets (see figure 9.7). For $G-$means and *dip*$-$means we show the results with two different parameter values. Indeed $X-$means has no parameter, $G-$means requires a $p-$value threshold for the Anderson-Darling test and *dip*$-$means needs two parameters (a $p-$value threshold for the dip-test and a majority level for a voting mechanism that we keep at 50%).

First we should notice the efficiency of the Elbow method. Even if the logic behind is not as smart as the other wrappers, its results are of good quality. Second, we can observe the general drawback of gaussian based algorithm ($X-$means and $G-$means): they tend to over-cluster when data are not gaussian-shaped (see Triangles, Chevrons and No structure examples). *dip*$-$means is naturally more robust to these geometry since it is also based on the unimodality assumption of the clusters. In comparison with $\Phi-$means, it better treats the Anisotropic case. Nonetheless, *dip*$-$means is less efficient in the Several blobs and Triangles cases.

Real-world datasets Here we compare the clustering algorithm on real-world datasets. For every algorithm we try to tune the parameters so as to get the most relevant results.

Let us start with the famous `Iris` dataset which contains 150 instances of iris plants described by 4 attributes ($n = 150, d = 4$) and classified into three classes. The table 9.2 presents the results of the clustering algorithms. Here we may recall that the classes are not well-separated: two of the three classes are rather close in the feature space. This can explain why Elbow, $G-$means and *dip*$-$means output 2 clusters (their clusters are also the same). $X-$means clearly over-clusters the dataset while $\Phi-$means outputs the right number of clusters. Moreover the clusters output by $\Phi-$means are rather close the initial classes (accuracy equal to 0.893).

Algorithm	Number of found clusters
Elbow	2
$G-$means (10^{-3})	2
$X-$means	12
dip$-$means (0.05)	2
$\Phi-$means (10^{-30})	3

Table 9.2: *Number of clusters found after parameter optimization on* `Iris` *dataset*

`Digits` is an optical recognition of handwritten digits dataset which gathers 5620 8x8 images of integer pixels in the range 0..16 ($n = 5620, d = 64$). There are naturally 10 classes (from 0 to 9). As a classical pre-processing step, we reduce the dimension with PCA to 28, which corresponds to the 95% of explained variance. The table 9.3 shows the results of the algorithms. Once again, Elbow is able to provide good results. By tuning $G-$means we can get the right number of classes. $\Phi-$means outputs rather the same thing for a wide range of parameter values and gives the same result as $X-$means does. Finally, we do not manage to get more reasonable results with *dip*$-$means.

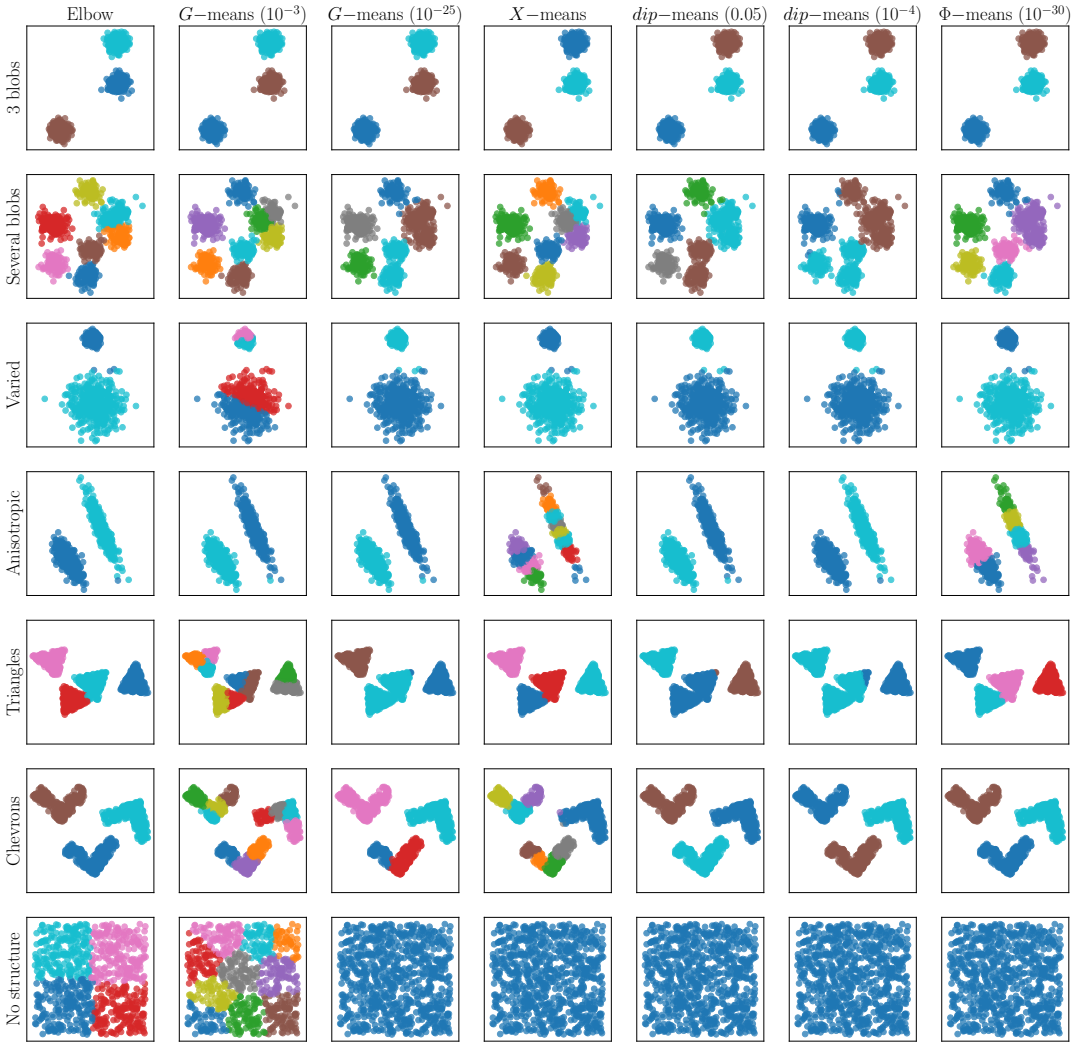


Figure 9.7: *Comparison of $k-$means wrappers on 2D toy datasets. Parameter values are given in brackets.*

Algorithm	Number of found clusters
Elbow	9
$G-$means (10^{-25})	10
$X-$means	16
dip$-$means (0.05)	1
$\Phi-$means (10^{-30})	16

Table 9.3: *Number of clusters found after parameter optimization on* `Digits` *dataset*

The `Credit Card`[3] dataset gathers 17 behavioral variables of about 9000 active credit card holders ($n = 8636, d = 17$ after removing incomplete records). This dataset aims to develop a customer segmentation to define marketing strategy. The features being rather imbalanced we standardized the data by removing the mean and scaling to unit variance. The results of the table 9.4 show that the wrappers rather agree with the Elbow method. Once again, we did not manage to get better results with *dip*−means.

Algorithm	Number of found clusters
Elbow ($k_{\max} = 30$)	24
G−means (10^{-3})	29
G−means (10^{-25})	26
X−means	27
dip−means	1
Φ−means (10^{-30})	27

Table 9.4: *Number of clusters found after parameter optimization on* `Credit Card` *dataset*

Time performances Here we compare the speed of the wrappers according to the number of observations. In this experiment, we run each algorithm on 250 different 3-clusters datasets in dimension $d = 3$ (varying the number of observations n). These runs have been performed on a single core Intel(R) Xeon(R) E5-2695 v4 at 2.10GHz. The figure 9.8 shows the average running time of all the wrappers. In particular, we observe that *dip*−means is far more complex than the others, even the Elbow method (for which we keep $k_{\max} = 15$). We recall that *dip*−means need to compute pairwise distances, so it has $O(n^2)$ complexity while other algorithm linearly depend on the number of observations. As an example, *dip*−means already takes more than 10s to process 1000 records (in dimension 3), so unlike G−means, X−means and Φ−means, it is not scalable.

k−means usage Finally, we want to highlight one important feature of Φ−means: the low usage of k−means. On the figure 9.9, we have plotted the number of k−means iterations requested to perform the clustering for every algorithm. Here we have generated 100 k−clusters configurations ($n = 900, d = 4$) with $k = 3, 6$ and 9 and we retrieved the average number of k−means iterations (the standard deviation is also shown on the figure).

First of all, we may wonder why the Elbow method reduces its number of k−means iterations when the number of clusters increases. One reason would be the following: when the real number of clusters is low but k rather high, it needs more iterations because there is more concurrency between the centers. Otherwise, we clearly notice that

[3] `https://www.kaggle.com/arjunbhasin2013/ccdata`

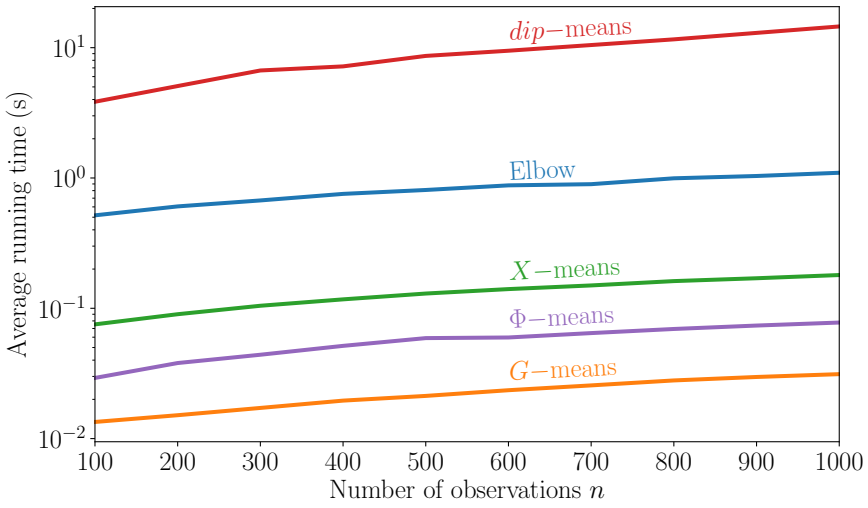


Figure 9.8: *Performances comparison of the k−means wrappers according to the number of observations n (3 clusters)*

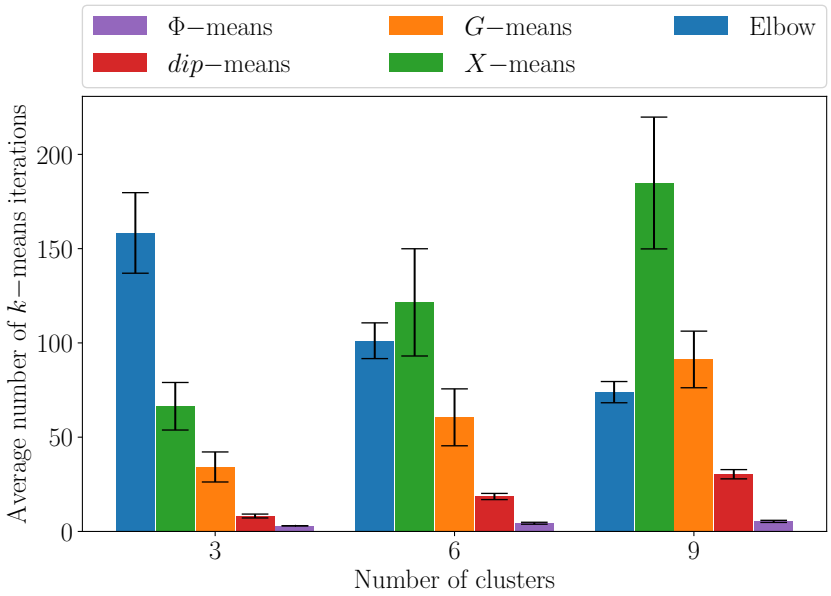


Figure 9.9: *Comparison of the k−means usage between wrappers. The bar heights gives the average number of iterations while the error bars represent the standard deviation*

Φ−means really does need many k−means iterations to perform a relevant clustering. Moreover, it is not so much impacted by the number of clusters. This phenomenon highlights the power of the split performed by Φ−means, so the power of the hyperplane

$\mathcal{H}^*$. Indirectly, it is also a reason explaining the good time performances of the algorithm.

9.5 Conclusion and perspective

In this last chapter, we have given another view of the problem solved by HFTU. It lead us to design a new heuristic to compute the separating hyperplane $\mathcal{H}^*$. This heuristic scales better with the dimension and combined with few random initializations it can even provide better results than the legacy algorithm.

Based on this heuristic, we have developed $\Phi-$means, a clustering algorithm which incrementally find the k of $k-$means. Unlike the great majority of previous $k-$means wrappers, $\Phi-$means is based on the unimodality assumption of the clusters (UAC) which is far weaker than the common gaussian-like assumptions. It is also the first time such an algorithm has been linearly dependent on the number of observations. The *dip*$-$means algorithm also relies on UAC but with $O(n^2)$ complexity it cannot scale as other algorithms. Through our different experiments we highlight the behavior and the performances of $\Phi-$means. To our point of view, it is totally competitive with the other $k-$means-based clustering algorithms but with several key assets: it can tackle a wider variety of cluster shapes and it does not heavily depends on $k-$means. Today, we conceive that $\Phi-$means is not totally mature. Some tests and experiments could naturally be further developed.

About the perspectives, the heuristic to compute HFTU opens new possibilities. First, as we said before, the heuristic iterates like $k-$means so it is likely to work on streaming data through a sliding window model. Let us consider a window W at a given time where HFTU is performed. When the window moves, the new data is added and the oldest is removed. We just have to perform very few iterations of the heuristic to update $\mathcal{H}^*$ (and perform the test). Second, we think that the problem solved by HFTU can be *kernelized*, meaning that it could be possible to compute a non-linear boundary between data, analog to $\mathcal{H}^*$ in the classical HFTU. This transformation does call into question the relevance of the unimodality test since the distribution is not the same in the space spanned by the kernel (RKHS). Nonetheless, if we only focus on the unsupervised splitting property of $\mathcal{H}^*$, the "kernelization" could provide a relevant boundary in a wider variety of data configurations.

Conclusion

In this thesis, we have presented new statistical methods for data mining. This work aims to encompass both theoretical and practical aspects insofar as it shows the relevance of our contributions. The most powerful theory should solve the most practical real-world issues, so it was essential for us to go all the way.

In the first part, we mainly presented SPOT, a new statistical anomaly detector able to find outliers in univariate streaming data without distribution assumption. While previous statistical approaches were limited either by the distribution choice (parametric techniques) or the restricted resolution of the data-driven approaches (non-parametric techniques), SPOT combines their respective advantages: it is based on a very generic model (based on Extreme Value Theory) which easily explores the low density regions. We have shown that SPOT (and its variant DSPOT) can be used in several contexts whether to detect anomalies or more generally to perform automatic thresholding.

As a direct concrete application, we embeds SPOT in a simple but powerful intrusion detection system, called `netspot`. Aware of the many challenges and requirements regarding the use of AI in cyber security, `netspot` approaches the issues from a different angle: simple design, few parameters, fast processing with a balance between expertise and algorithmic. This *downsized* architecture allows however to detect real-world cyber-attacks.

In the second part, we have been focused on a less disclosed research area: unimodality testing. We detail two new unimodality tests we developed, FTU and HFTU, which are based on the same univariate mechanism but with different higher-dimensional generalizations. Compared to the literature, the Folding Test of

Unimodality (FTU) makes two breakthroughs: it can tackle multivariate data and also streaming data. These features open the range of possibilities to exploit unimodality testing. In line with this work, we have developed the Hyperplane Folding Test of Unimodality (HFTU) so as to solve some limits of FTU.

Afterwards, the main asset of HFTU is the hyperplane it builds during the folding mechanism. We use the properties of this hyperplane to design a new clustering algorithm: $\Phi-$means. This algorithm incrementally finds the right k of $k-$means based on the unimodality assumption of the clusters which is less restrictive than the common gaussian-based techniques. Our experiments have shown that $\Phi-$means is comparable with the state-of-the-art and yet far more generic.

Moreover, we have also worked to make our contributions really usable for all, distributing all our code through turnkey packages. We hope that it could help the research and even a wider community.

Open problems Despite these advances many challenges naturally remain and below we detail some which are directly related to this thesis.

- The main limit of the SPOT algorithm is its inability to tackle multivariate data. The reason is that the extreme value theory does not generalizes "well" to higher dimensions. Indeed, we can easily understand what is an *extreme value* in the univariate case, but it is harder to interpret it in the multivariate case. The multivariate extreme value theory aims to build a model in the joint extreme zone (where all the components are extreme). However, low-probability regions do not limit to this area and we rather would like such a generic model for extreme probability density contours. We explored the use of copulas but the minimum volume sets (MV-sets) approach seems to be promising [123].

- Our experiments have shown that FTU and HFTU are able to discriminate unimodal distributions from multimodal ones. In the univariate case, we have performed several analytical checks on common distributions but we have not a general proof showing that the tests are true. As FTU uses approximations, we rather think that we can prove some properties for HFTU. One first possible result would be the following sufficient condition: let X be an unimodal real random variable (twice integrable), then $\Phi_{\text{HFTU}}(X) \geq 1$. Indeed, unimodal random variable have several properties and characterizations that could be exploited to show such a result (see the reference [24]). This result in the univariate case is also likely to be generalized to higher dimensions.

- In the longer term, we think that statistics can be introduced to other data mining sub-fields where classical algorithmic approaches are limited. For instance, the pattern mining aims to find structures in the input data however it generally suffers from the huge amount of output patterns. In this particular case, statistics have been introduced so as to filter the results (*statistically sound pattern discovery*). The fundamental idea is to remove "patterns that are found

in the sample data but do not hold in the wider population from which the sample was drawn" [49]. Here, we clearly observe the balance between data mining, which focuses on the samples, and statistics, which estimate whether a sample is obtained from an underlying generative process. The approaches are mainly based on multiple hypothesis testing procedures to keep the most *significant* patterns (see [68]).

Appendices

A

FTU materials

A.1 Preliminaries

This appendix deals with some mathematical results about the folding values so as to ease the proofs given in appendix A.2. It aims to be self-contained, so not depending on results given in the main part of the document.

In the general case, X defines a random column vector of $\mathbb{R}^d$ ($d \in \mathbb{N}^*$). We note Σ its covariance matrix

$$\Sigma = \mathrm{Cov}(X, X) = \mathbb{E}(XX^T) - \mathbb{E}(X)\,\mathbb{E}(X)^T$$

This matrix is supposed definite-positive so invertible.

DEFINITION A.1 (Folding pivot) *Let X be a $3-$integrable random variables. We define its* ***folding pivot*** *as*

$$s^*(X) = \frac{1}{2}\,\Sigma^{-1}\,\mathrm{Cov}\left(X, \|X\|^2\right).$$

DEFINITION A.2 (Folding ratio) *Let X be a $3-$integrable random variables. We define its* ***folding ratio*** *as*

$$\phi(X) = \frac{\mathrm{Var}\,\|X - s^*\|}{\mathrm{Tr}\,\Sigma}.$$

REMARK A.1 *In the univariate case ($d = 1$), the previous formulas have simpler expressions:*

$$s^*(X) = \frac{1}{2}\frac{\mathrm{Cov}(X, X^2)}{\mathrm{Var}\, X} \qquad \textit{and} \qquad \phi(X) = \frac{\mathrm{Var}\,|X - s^*|}{\mathrm{Var}\, X}$$

PROPOSITION A.1 *Let X be a $3-$integrable random vector of $\mathbb{R}^n$. For all $\lambda \in \mathbb{R}^*, b \in \mathbb{R}^n$, we have:*

$$s^*(\lambda X + b) = \lambda s^*(X) + b$$

Proof. First, we have $\Sigma(\lambda X + b) = \mathrm{Cov}(\lambda X + b, \lambda X + b) = \lambda^2\, \Sigma(X)$. As $\lambda \neq 0$, we also get $\Sigma^{-1}(\lambda X + b) = \lambda^{-2}\, \Sigma^{-1}(X)$. Moreover,

$$\begin{aligned}
\mathrm{Cov}\left(\lambda X + b, \|\lambda X + b\|^2\right) &= \lambda\, \mathrm{Cov}\left(X, \lambda^2 \|X\|^2 + 2\lambda b^T X\right) \\
&= \lambda^3\, \mathrm{Cov}\left(X, \|X\|^2\right) + 2\lambda^2\, \mathrm{Cov}\left(X, b^T X\right) \\
&= \lambda^3\, \mathrm{Cov}\left(X, \|X\|^2\right) + 2\lambda^2\, \Sigma(X) b.
\end{aligned}$$

Finally,

$$\begin{aligned}
s^*(\lambda X + b) &= \frac{1}{2}\lambda^{-2}\, \Sigma^{-1}(X) \left(\lambda^3\, \mathrm{Cov}\left(X, \|X\|^2\right) + 2\lambda^2\, \Sigma(X) b\right) \\
&= \frac{1}{2}\lambda\, \Sigma(X)\, \mathrm{Cov}\left(X, \|X\|^2\right) + b \\
&= \lambda s^*(X) + b
\end{aligned}$$

□

PROPOSITION A.2 *Let X be a $3-$integrable random vector of $\mathbb{R}^n$. For all $\lambda \in \mathbb{R}^*, b \in \mathbb{R}^n$, we have:*

$$\phi(\lambda X + b) = \phi(X)$$

Proof. We know that $\Sigma(\lambda X + b) = \lambda^2\, \Sigma(X)$ so $\mathrm{Tr}\left(\Sigma(\lambda X + b)\right) = \lambda^2\, \mathrm{Tr}\, \Sigma$. With the previosu result we have $s^*(\lambda X + b) = \lambda s^*(X) + b$ so

$$\begin{aligned}
\phi(\lambda X + b) &= \frac{\mathrm{Var}\,|\lambda X + b - \lambda s^*(X) - b|}{\lambda^2\, \mathrm{Tr}\, \Sigma} \\
&= \frac{\lambda^2\, \mathrm{Var}\,|X - s^*(X)|}{\lambda^2\, \mathrm{Tr}\, \Sigma} = \phi(X)
\end{aligned}$$

□

PROPOSITION A.3 *Let X be a square integrable real random variable and $s \in \mathbb{R}$. We have the following identity:*

$$\mathrm{Var}\,|X - s| = \mathrm{Var}(X) + \left(\mathbb{E}(X) - s\right)^2 - \left(\mathbb{E}\,|X - s|\right)^2$$

Proof.

$$\begin{aligned}\operatorname{Var}|X-s| &= \mathbb{E}\left(|X-s|^2\right) - \left(\mathbb{E}|X-s|\right)^2 \\ &= \operatorname{Var}(X-s) + \left(\mathbb{E}(X-s)\right)^2 - \left(\mathbb{E}|X-s|\right)^2 \\ &= \operatorname{Var}(X) + \left(\mathbb{E}(X)-s\right)^2 - \left(\mathbb{E}|X-s|\right)^2\end{aligned}$$

□

REMARK A.2 *In particular, if* $s = \mathbb{E}(X)$ *we have* $\operatorname{Var}|X-s| = \operatorname{Var}(X) - \left(\mathbb{E}|X-s|\right)^2$

LEMMA A.1 *If* X *is symmetrical about* 0 *and* $\forall k \in \mathbb{N}, \mathbb{E}(|X|^{2k+1})$ *exists, then* $\mathbb{E}(X^{2k+1}) = 0$

Proof. Let us suppose X symmetrical about zero. It means that its density is even. So,

$$\begin{aligned}\mathbb{E}(X^{2k+1}) = \int_{\mathbb{R}} x^{2k+1} f(x)\,\mathrm{d}x &= \int_{-\infty}^{0} x^{2k+1} f(x)\,\mathrm{d}x + \int_{0}^{+\infty} x^{2k+1} f(x)\,\mathrm{d}x \\ &= \int_{0}^{+\infty} (-x)^{2k+1} f(-x)\,\mathrm{d}x + \int_{0}^{+\infty} x^{2k+1} f(x)\,\mathrm{d}x \\ &= 0\end{aligned}$$

□

REMARK A.3 *More generally, if* X *is symmetrical about* $m \in \mathbb{R}$*, then*

$$\mathbb{E}\left((X-m)^{2k+1}\right) = 0$$

PROPOSITION A.4 *If the density of* X *is symmetrical about* $m \in \mathbb{R}$ *then* $\mathbb{E}(X) = s^* = m$*.*

Proof. Let us consider the density f symmetrical about m. With the previous result we have $\mathbb{E}(X-m) = 0$ so $\mathbb{E}(X) = m$. Now, let us consider the centred random variable $Y = X - \mathbb{E}(X) = X - m$. With the previous result we also have $\mathbb{E}(Y^3) = 0$, then

$$\operatorname{Cov}(Y, Y^2) = \mathbb{E}(Y^3) - \mathbb{E}(Y)\,\mathbb{E}(Y^2) = 0$$

But,

$$\begin{aligned}\operatorname{Cov}(Y, Y^2) &= \operatorname{Cov}\left(X-m, (X-m)^2\right) \\ &= \operatorname{Cov}\left(X-m, X^2 - 2mX + m^2\right) \\ &= \operatorname{Cov}(X, X^2 - 2mX) \\ &= \operatorname{Cov}(X, X^2) - 2m\operatorname{Var}X\end{aligned}$$

So it means that $\operatorname{Cov}(X, X^2) = 2m\operatorname{Var}X$. Finally,

$$s^* = \frac{1}{2}\frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var}X} = m$$

□

PROPOSITION A.5 *Let X be a random variable with cdf F and density f such that its folding pivot s^* is well defined. Then*

$$\mathbb{E}\,|X - s^*| = s^*\,(2F(s^*) - 1) + \mathbb{E}(X) - 2\int_{-\infty}^{s^*} x f(x)\,\mathrm{d}x.$$

Proof. Under the given hypothesis:

$$\begin{aligned}
\mathbb{E}\,|X - s^*| &= \int_{\mathbb{R}} |x - s^*|\, f(x)\,\mathrm{d}x \\
&= \int_{-\infty}^{s^*} (s^* - x) f(x)\,\mathrm{d}x + \int_{s^*}^{+\infty} (x - s^*) f(x)\,\mathrm{d}x \\
&= s^* F(s^*) - \int_{-\infty}^{s^*} x f(x)\,\mathrm{d}x + \int_{s^*}^{+\infty} x f(x)\,\mathrm{d}x - s^*(1 - F(s^*)) \\
&= s^*\,(2F(s^*) - 1) + \mathbb{E}(X) - 2\int_{-\infty}^{s^*} x f(x)\,\mathrm{d}x
\end{aligned}$$

□

COROLLARY A.1 *In addition we suppose X symmetrical. Then*

$$\mathbb{E}\,|X - s^*| = \mathbb{E}(X) - 2\int_{-\infty}^{\mathbb{E}(X)} x f(x)\,\mathrm{d}x = 2\int_{\mathbb{E}(X)}^{\infty} x f(x)\,\mathrm{d}x - \mathbb{E}(X).$$

A.2 Folding computation of common distributions

This section shows the calculations of the folding pivot and folding ratios of some common univariate distributions. Unlike appendix A.1, here we use a less formal approach to show the results (no proposition, theorem, lemma etc.) to make it more flowing.

Two point masses Here we suppose that X follows the following discrete distribution:

X	0	d
$\mathbb{P}(X)$	α	$1-\alpha$

This distribution is not symmetrical but actually the pivot is not impacted by the asymmetry of the distribution. Indeed we have $\mathbb{E}(X^k) = (1-\alpha)d^k$ (and $\operatorname{Var} X = \alpha(1-\alpha)d^2$), then

$$\operatorname{Cov}(X, X^2) = \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2) = \alpha(1-\alpha)d^3,$$

and

$$s^* = \frac{1}{2}\frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \frac{1}{2}\frac{\alpha(1-\alpha)d^3}{\alpha(1-\alpha)d^2} = \frac{d}{2}$$

As s^* is in the middle of the two point masses, we also have $\mathbb{E}\,|X - s^*| = \dfrac{d}{2}$, so

$$\begin{aligned}
\operatorname{Var}|X - s^*| &= \operatorname{Var}(X) + (\mathbb{E}(X) - s^*)^2 - (\mathbb{E}\,|X - s^*|)^2 \\
&= \alpha(1-\alpha)d^2 + \left((1-\alpha)d - \frac{d}{2}\right)^2 - \left(\frac{d}{2}\right)^2 \\
&= \alpha(1-\alpha)d^2 + (1-\alpha)d\,((1-\alpha)d - d) \\
&= 0
\end{aligned}$$

So we have $\phi(X) = 0$.

Normal distribution Here we suppose that $X \sim \mathcal{N}(\mu, \Sigma^2)$. With the previous results, we know that $s^* = \mathbb{E}(X) = \mu$ and then

$$\operatorname{Var}|X - s^*| = \operatorname{Var}(X) - (\mathbb{E}\,|X - s^*|)^2$$

As we know the first term, computing the folding ratio stems down to compute $\mathbb{E}\,|X - s^*|$. In our case

$$\begin{aligned}
\mathbb{E}\,|X - s^*| = \int_{\mathbb{R}} |x - \mu|\, f(x)\, \mathrm{d}x &= \int_{\mathbb{R}} |x|\, f(x + \mu)\, \mathrm{d}x \\
&= 2\int_0^{+\infty} x f(x + \mu)\, \mathrm{d}x \\
&= 2\int_0^{+\infty} x \frac{1}{\Sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\,\Sigma^2}\right) \mathrm{d}x \\
&= \frac{-2\,\Sigma^2}{\Sigma\sqrt{2\pi}} \left[\exp\left(-\frac{x^2}{2\,\Sigma^2}\right)\right]_0^{+\infty} = \frac{2\,\Sigma}{\sqrt{2\pi}}
\end{aligned}$$

So we have

$$\operatorname{Var}|X - s^*| = \Sigma^2 - \left(\frac{2\,\Sigma}{\sqrt{2\pi}}\right)^2 = \Sigma^2\left(1 - \frac{2}{\pi}\right)$$

Finally

$$\phi(X) = \frac{\operatorname{Var}|X - s^*|}{\operatorname{Var} X} = 1 - \frac{2}{\pi}$$

Laplace distribution Here we suppose that $X \sim \mathcal{L}(\mu, b)$. We recall the density of the Laplace distribution:

$$f(x) = \frac{1}{2b} \exp\left(-\frac{|x - \mu|}{b}\right)$$

We have $\mathbb{E}(X) = \mu$ and $\operatorname{Var} X = 2b^2$. It is also a symmetrical distribution so we especially have to compute $\mathbb{E}\,|X - s^*| = \mathbb{E}\,|X - \mu|$.

$$\begin{aligned}
\mathbb{E}\,|X-\mu| &= \frac{1}{2b}\int_{\mathbb{R}} |x-\mu| \exp\left(-\frac{|x-\mu|}{b}\right)\,\mathrm{d}x \\
&= \frac{1}{2b}\int_{-\infty}^{\mu} -(x-\mu)\exp\left(\frac{x-\mu}{b}\right)\,\mathrm{d}x + \frac{1}{2b}\int_{\mu}^{+\infty}(x-\mu)\exp\left(-\frac{x-\mu}{b}\right)\,\mathrm{d}x \\
&= \frac{1}{b}\int_0^{\infty} t\exp\left(-\frac{t}{b}\right)\,\mathrm{d}t \\
&= b\int_0^{\infty} t\mathrm{e}^{-t}\,\mathrm{d}t = b
\end{aligned}$$

So we get $\operatorname{Var}|X - s^*| = \operatorname{Var} X - (\mathbb{E}\,|X-\mu|)^2 = b^2$ and the folding ratio is $\phi(X) = \dfrac{1}{2}$.

Arcsine distribution Here we suppose that X follow the arcsine distribution on bounded support $[a, b]$. We recall its density:

$$f(x) = \frac{1}{\pi\sqrt{(x-a)(b-x)}}$$

We have $\mathbb{E}(X) = \frac{a+b}{2}$ and $\operatorname{Var} X = \frac{1}{8}(b-a)^2$. It is also a symmetrical distribution but with two modes (a and b). Naturally $s^* = \mathbb{E}(X) = \frac{a+b}{2}$ and we compute:

$$\begin{aligned}
\mathbb{E}\,|X-s^*| &= \frac{1}{\pi}\int_a^b \left|x - \frac{a+b}{2}\right| \frac{1}{\sqrt{(x-a)(b-x)}}\,\mathrm{d}x \\
&= \frac{1}{2\pi}\int_a^b |2x-a-b|\,\frac{1}{\sqrt{(x-a)(b-x)}}\,\mathrm{d}x
\end{aligned}$$

The integrand is clearly symmetrical about $\frac{a+b}{2}$ then

$$\mathbb{E}\,|X-s^*| = \frac{1}{\pi}\int_{\frac{a+b}{2}}^b \frac{2x-a-b}{\sqrt{(x-a)(b-x)}}\,\mathrm{d}x$$

A primitive of $x \mapsto \dfrac{2x-a-b}{\sqrt{(x-a)(b-x)}}$ is $x \mapsto -2\sqrt{(x-a)(b-x)}$ so

$$\begin{aligned}
\mathbb{E}\,|X-s^*| &= \frac{1}{\pi}\left[-2\sqrt{(x-a)(b-x)}\right]_{\frac{a+b}{2}}^{b} \\
&= \frac{b-a}{\pi}
\end{aligned}$$

Then

$$\begin{aligned}
\phi(X) = \frac{\operatorname{Var} X - (\mathbb{E}\,|X-s^*|)^2}{\operatorname{Var} X} &= 1 - \frac{(\mathbb{E}\,|X-s^*|)^2}{\operatorname{Var} X} \\
&= 1 - \frac{8}{\pi^2}
\end{aligned}$$

Exponential distribution Let $\lambda > 0$, we suppose $X \sim \mathcal{E}(\lambda)$. The density of X is (for all $x \geq 0$):

$$f(x) = \lambda \exp(-\lambda x).$$

The density is not symmetrical, so we have to compute the pivot s^*. We recall its definition (in the univariate case):

$$s^* = \frac{1}{2} \frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X}$$

We already have $\operatorname{Var} X = \frac{1}{\lambda^2}$ so we compute the covariance.

$$\operatorname{Cov}(X, X^2) = \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2)$$

Actually the moments of the distribution are given by $\forall k \in \mathbb{N}^*, \mathbb{E}(X^k) = \dfrac{k!}{\lambda^k}$. Then

$$\operatorname{Cov}(X, X^2) = \frac{6}{\lambda^3} - \frac{1}{\lambda} \cdot \frac{2}{\lambda^2} = \frac{4}{\lambda^3},$$

and

$$s^* = \frac{1}{2} \frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \frac{2}{\lambda}$$

Now, we need to compute

$$\begin{aligned}
\mathbb{E}\,|X - s^*| &= \lambda \int_0^\infty \left| x - \frac{2}{\lambda} \right| \exp(-\lambda x)\ \mathrm{d}x \\
&= \int_0^\infty |\lambda x - 2| \exp(-\lambda x)\ \mathrm{d}x \\
&= \frac{1}{\lambda} \int_0^\infty |t - 2| \exp(-t)\ \mathrm{d}t \\
&= \frac{1}{\lambda} \int_0^2 (2 - t) \exp(-t)\ \mathrm{d}t - \frac{1}{\lambda} \int_2^\infty (2 - t) \exp(-t)\ \mathrm{d}t.
\end{aligned}$$

A primitive of $t \mapsto t \exp(-t)$ is $t \mapsto -(1+t)\exp(-t)$ so a primitive of $t \mapsto (2-t)\exp(-t)$ is $t \mapsto (t-1)\exp(-t)$. We get

$$\begin{aligned}
\mathbb{E}\,|X - s^*| &= \frac{1}{\lambda} \left([(t-1)\exp(-t)]_0^2 - [(t-1)\exp(-t)]_2^\infty \right) \\
&= \frac{1}{\lambda} \left(e^{-2} + 1 + e^{-2} \right) = \frac{1}{\lambda} \left(1 + 2e^{-2} \right).
\end{aligned}$$

So for the variance we have

$$\begin{aligned}
\operatorname{Var}|X - s^*| &= \operatorname{Var}(X) + (\mathbb{E}(X) - s^*)^2 - (\mathbb{E}\,|X - s^*|)^2 \\
&= \frac{1}{\lambda^2} + \left(\frac{1}{\lambda} - \frac{2}{\lambda} \right)^2 - \frac{1}{\lambda^2} \left(1 + 2e^{-2} \right)^2 \\
&= \frac{1}{\lambda^2} \left(2 - \left(1 + 2e^{-2} \right)^2 \right) \\
&= \frac{1}{\lambda^2} \left(1 - 4e^{-2} - 4e^{-4} \right).
\end{aligned}$$

Finally $\phi(X) = 1 - 4e^{-2} - 4e^{-4}$.

χ^2 distribution Let $k \in \mathbb{N}^*$, we assume that X follows a chi-squared distribution with k degrees of freedom. Its density is defined for all $x > 0$ by:

$$f(x) = \frac{x^{\frac{k}{2}-1} \exp\left(-\frac{x}{2}\right)}{2^{\frac{k}{2}}\Gamma\left(\frac{k}{2}\right)}$$

First we can recall that for all $n \in \mathbb{N}^*$ we have

$$\mathbb{E}(X^n) = \prod_{i=0}^{n-1}(k+2i),$$

meaning that $\mathbb{E}(X) = k, \mathbb{E}(X^2) = k(k+2)$ and $\mathbb{E}(X^3) = k(k+2)(k+4)$ (we also have $\operatorname{Var} X = 2k$). Then

$$\operatorname{Cov}(X, X^2) = \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2) = k(k+2)(k+4) - k^2(k+2) = 4k(k+2).$$

Finally,

$$s^* = \frac{1}{2}\frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \frac{4k(k+2)}{4k} = k+2$$

Gamma distribution We consider that X follows a gamma distribution with parameters k, θ those density is defined on $]0, +\infty[$ by

$$f(x) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} \exp\left(-\frac{x}{\theta}\right).$$

We know that for all $n \in \mathbb{N}^*$,

$$\mathbb{E}(X^n) = \theta^n \frac{(n+k-1)!}{(k-1)!}.$$

We have

$$\begin{aligned}\operatorname{Cov}(X, X^2) &= \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2) \\ &= \theta^3 k(k+1)(k+2) - \theta k \cdot \theta^2 k(k+1) \\ &= 2k(k+1)\theta^3.\end{aligned}$$

As $\operatorname{Var} X = k\theta^2$, the pivot is then given by

$$s^* = \frac{1}{2}\frac{2k(k+1)\theta^3}{k\theta^2} = (k+1)\theta.$$

Beta distribution We consider that X follows a beta distribution with parameters α, β those density is defined on $[0, 1]$ (or $]0, 1[$) by

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathrm{B}(\alpha, \beta)}$$

with $\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$.

General case

We know that for all $k \in \mathbb{N}^*$,

$$\mathbb{E}(X^k) = \prod_{i=0}^{k-1} \frac{\alpha + i}{\alpha + \beta + i}.$$

So it notably gives

$$\mathbb{E}(X) = \frac{\alpha}{\alpha + \beta} \quad \mathbb{E}(X^2) = \frac{\alpha(\alpha+1)}{(\alpha+\beta)(\alpha+\beta+1)} \quad \mathbb{E}(X^3) = \frac{\alpha+2}{\alpha+\beta+2}\,\mathbb{E}(X^2).$$

We can compute

$$\begin{aligned}
\operatorname{Cov}(X, X^2) &= \mathbb{E}(X^3) - \mathbb{E}(X)\,\mathbb{E}(X^2) \\
&= \mathbb{E}(X^2)\left(\frac{\alpha+2}{\alpha+\beta+2} - \frac{\alpha}{\alpha+\beta}\right) \\
&= \mathbb{E}(X^2)\frac{2\beta}{(\alpha+\beta)(\alpha+\beta+2)} \\
&= \frac{2\alpha\beta(\alpha+1)}{(\alpha+\beta)^2(\alpha+\beta+1)(\alpha+\beta+2)}
\end{aligned}$$

As $\operatorname{Var} X = \dfrac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$ we can compute the pivot:

$$s^* = \frac{1}{2}\frac{\operatorname{Cov}(X, X^2)}{\operatorname{Var} X} = \frac{\alpha+1}{\alpha+\beta+2}$$

Symmetrical case $\alpha = \beta$

Unfortunately the direct calculation of the folding ratio is not easy because of the expression of $\mathbb{E}\,|X - s^*|$ which is not really explicit. However we can limit the study to the symmetrical case $\alpha = \beta$. In this case the density is symmetrical about $\frac{1}{2}$ which an interesting evolving shape with α. When $\alpha < 1$, the density has a U shape (two modes in 0 and 1). For example, when $\alpha = \frac{1}{2}$, we get the arcsine distribution. The case $\alpha = 1$ corresponds to the uniform density and for $\alpha > 1$ with have a unimodal density. Actually we can check that:

$$s^* = \frac{\alpha+1}{\alpha+\beta+2} = \frac{1}{2}.$$

Let us calculate the *folded* expected value:

$$\begin{aligned}
\mathbb{E}\,|X_\alpha - s^*| &= \frac{1}{\mathrm{B}(\alpha,\alpha)} \int_0^1 \left|x - \frac{1}{2}\right| (x(1-x))^{\alpha-1}\ \mathrm{d}x \\
&= \frac{2}{\mathrm{B}(\alpha,\alpha)} \int_0^{\frac{1}{2}} \left(\frac{1}{2} - x\right) (x(1-x))^{\alpha-1}\ \mathrm{d}x \\
&= \frac{1}{\mathrm{B}(\alpha,\alpha)} \int_0^{\frac{1}{2}} (1-2x)\,(x(1-x))^{\alpha-1}\ \mathrm{d}x.
\end{aligned}$$

A primitive of $x \mapsto (1-2x)\,(x(1-x))^{\alpha-1}$ is $x \mapsto \dfrac{1}{\alpha}\,(x(1-x))^{\alpha}$ so

$$\begin{aligned}\mathbb{E}\,|X_\alpha - s^*| &= \frac{1}{\mathrm{B}(\alpha,\alpha)}\left[\frac{1}{\alpha}\,(x(1-x))^{\alpha}\right]_0^{\frac{1}{2}} \\ &= \frac{1}{\alpha 4^{\alpha}\,\mathrm{B}(\alpha,\alpha)}.\end{aligned}$$

With the following identity

$$\mathrm{B}\,(\alpha,\alpha) = \frac{2}{4^{\alpha}}\,\mathrm{B}\left(\frac{1}{2},\alpha\right),$$

we finally get

$$\mathbb{E}\,|X_\alpha - s^*| = \frac{1}{2\alpha\,\mathrm{B}\left(\frac{1}{2},\alpha\right)}.$$

For the folded ratio we then have

$$\begin{aligned}\phi(X_\alpha) &= 1 - \frac{(\mathbb{E}\,|X_\alpha - s^*|)^2}{\operatorname{Var} X_\alpha} \\ &= 1 - \frac{2\alpha+1}{\alpha^2\,\mathrm{B}\left(\frac{1}{2},\alpha\right)^2}\end{aligned}$$

With the properties of the beta function, we can show that $\phi(X_\alpha) \underset{\alpha\to 0}{\longrightarrow} 0$ and $\phi(X_\alpha) \underset{\alpha\to+\infty}{\longrightarrow} 1-\frac{2}{\pi}$. The latter result is quite interesting because it corresponds to the folding ratio of the normal distribution. Actually, the beta density as a shape close to the gaussian density when $\alpha \gg 1$.

Furthermore, we can notice two things:

- $\phi(X_1) = \dfrac{1}{4}$ which is exactly the result of the uniform case;
- $\phi(X_{\frac{1}{2}}) = 1 - \dfrac{8}{\pi^2}$ which is exactly the result of the arcsine case.

Asymmetrical three point masses Let $a > 0$ and $\varepsilon \in \mathbb{R}$ such that $|\varepsilon| \leq a$. We suppose that X follows the following discrete distribution:

X	$-a$	ε	a
$\mathbb{P}(X)$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

This distribution is not symmetrical. First we need to calculate its basic moments.

$$\begin{aligned}
\mathbb{E}(X) &= \frac{1}{3}\left(-a+\varepsilon+a\right) = \frac{\varepsilon}{3}\\
\mathbb{E}(X^2) &= \frac{1}{3}\left((-a)^2+\varepsilon^2+a^2\right) = \frac{1}{3}\left(\varepsilon^2+2a^2\right)\\
\mathbb{E}(X^3) &= \frac{1}{3}\left((-a)^3+\varepsilon^3+a^3\right) = \frac{\varepsilon^3}{3}\\
\operatorname{Var} X &= \frac{1}{3}\left(\varepsilon^2+2a^2\right) - \left(\frac{\varepsilon}{3}\right)^2 = \frac{1}{9}\left(3\varepsilon^2+6a^2-\varepsilon^2\right) = \frac{2}{9}\left(\varepsilon^2+3a^2\right)\\
\operatorname{Cov}(X,X^2) &= \frac{\varepsilon^3}{3} - \frac{\varepsilon}{9}\left(\varepsilon^2+2a^2\right) = \frac{\varepsilon}{9}\left(3\varepsilon^2-\varepsilon^2-2a^2\right) = \frac{2\varepsilon}{9}\left(\varepsilon^2-a^2\right)
\end{aligned}$$

So we can calculate the folding pivot:

$$s^* = \frac{1}{2}\frac{\frac{2\varepsilon}{9}\left(\varepsilon^2-a^2\right)}{\frac{2}{9}\left(\varepsilon^2+3a^2\right)} = \frac{1}{2}\,\varepsilon\,\frac{\varepsilon^2-a^2}{\varepsilon^2+3a^2}$$

In particular we can notice that the sign of s^* depends on the sign of ε.

$$s^*-\varepsilon = \frac{\varepsilon}{2(\varepsilon^2+3a^2)}\left(\varepsilon^2-a^2-2\left(\varepsilon^2+3a^2\right)\right) = \frac{-\varepsilon\left(\varepsilon^2+7a^2\right)}{2(\varepsilon^2+3a^2)}$$

In the general case we have:

$$\mathbb{E}\left|X-s^*\right| = \frac{1}{3}\left(\left|-a-s^*\right|+\left|\varepsilon-s^*\right|+\left|a-s^*\right|\right).$$

If we assume $\varepsilon \geq 0$ we have $-a \leq s^* \leq 0 \leq \varepsilon \leq a$, so

$$\mathbb{E}\left|X-s^*\right| = \frac{1}{3}\left(s^*+a+\varepsilon-s^*+a-s^*\right) = \frac{1}{3}\left(2a+\varepsilon-s^*\right).$$

And if we assume $\varepsilon \leq 0$ we have $-a \leq \varepsilon \leq 0 \leq s^* \leq a$, so

$$\mathbb{E}\left|X-s^*\right| = \frac{1}{3}\left(s^*+a+s^*-\varepsilon+a-s^*\right) = \frac{1}{3}\left(2a-\varepsilon+s^*\right).$$

We can sum up with

$$\mathbb{E}\left|X-s^*\right| = \frac{1}{3}\left(2a+\left|\varepsilon-s^*\right|\right)$$

If we use the expression of s^* we get

$$\mathbb{E}\left|X-s^*\right| = \frac{1}{6(\varepsilon^2+3a^2)}\left(4a(\varepsilon^2+3a^2)+|\varepsilon|\left(\varepsilon^2+7a^2\right)\right)$$

On the other hand,

$$\begin{aligned}
\mathbb{E}\left((X-s^*)^2\right) &= \mathbb{E}(X^2) - 2s^*\,\mathbb{E}(X) + (s^*)^2\\
&= \frac{1}{3}\left(\varepsilon^2+2a^2\right) - \frac{2\varepsilon}{3}s^* + (s^*)^2
\end{aligned}$$

For the folded variance, the calculus is quite heavy, so we limit when $\varepsilon \geq 0$ (we can get the case $\varepsilon < 0$ by mapping ε to $-\varepsilon$):

$$\text{Var}\,|X - s^*| = \frac{2}{9}(a-\varepsilon)^2 \, \frac{9a^4 - 3a^3\varepsilon + 4a^2\varepsilon^2 + a\varepsilon^3 + \varepsilon^4}{(3a^2+\varepsilon^2)^2}$$

Finally:

$$\phi(X) = (a-\varepsilon)^2 \, \frac{9a^4 - 3a^3\varepsilon + 4a^2\varepsilon^2 + a\varepsilon^3 + \varepsilon^4}{(3a^2+\varepsilon^2)^3}.$$

We can notice that the folding ratio depends only on $r = \frac{\varepsilon}{a}$:

$$\phi(X) = (1-r)^2 \, \frac{9 - 3r + 4r^2 + r^3 + r^4}{(3+r^2)^3}$$

A.3 Numerical quantiles

The following parts describe the methodology to infer the formula 7.2. In particular, we study the influence of every parameter on q, leading us to a general expression to compute q in the last paragraph.

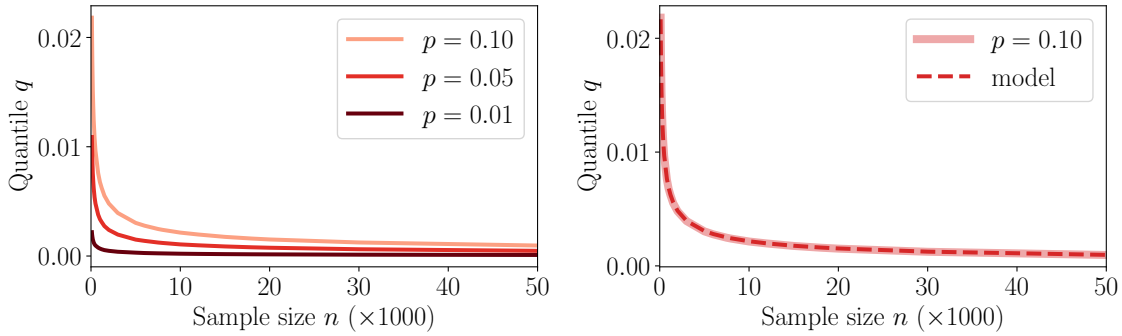
Sampling. To estimate q according to d, n, p, we have to generate (several times) n random points uniformly within the $d-$ball. This task is unfortunately not trivial, especially in high dimension (because of the curse of dimensionality). For this purpose we use the Muller's sampling method [28, 93] aimed to generate points uniformly over the surface of the $d-$ball (i.e. the $(d-1)-$sphere). Then thanks to the theorem A.1, we can reduce this sphere sampling to a ball sampling.

Algorithm 15 Muller's sampling

1: Sample d independent normally distributed variables $x_1, x_2 \dots x_d \sim \mathcal{N}(0,1)$
2: Let $\mathbf{X} = (x_1, x_2 \dots x_d)$. Compute $r = \|\mathbf{X}\|_2$.
3: **return** $(1/r) \times \mathbf{X}$

THEOREM A.1 *If $(s_1, s_2 \dots s_{d+2})$ is uniformly sampled over the surface of the $(d+2)-$ball then $(s_1, s_2 \dots s_d)$ is uniformly sampled within the $d-$ball.*

For every tuple (n, d) considered we generate $m = 10^6$ samples so as to estimate the cumulative distribution function of $Y_{n,d}$. In details, we experiment $d \in D = \{1, \dots 8\}$ with 25 different values of n between 100 and 50000 (we note N this set). For every tuple we retrieve the quantiles related to hundred $p-$values (we note P this set).



(a) *Influence of the sample size n on the quantile q for three p−values*

(b) *Comparison between the simulation and the model for $d = 3$ and $p = 0.1$*

Figure A.1: *Dependency on the sample size n*

Dependency in the sample size n. To show the dependence on n for the quantile q we take an example: $d = 3$ and $p \in \{0.25, 0.5, 0.75\}$. The behavior of q for this three $p-$values is presented on figure A.1a.

Naturally, the quantile decreases with the sample size. Indeed, when the sample size n is large, the folding statistics $\Phi(U_1 \ldots, U_n)$ will be close to its theoretical value (so $Y_{n,d}$ will be close to zero). Then for a given $p-$value, the quantile needed to make the decision will be tighter.

The shape of the curve lead us to model the dependence on n through $q(n, d, p) \propto \frac{1}{\sqrt{n}}$. To assess this dependence we evaluate the error made by such a model computed by least square regression. So, for all dimensions $d \in D$ and all $p-$values $p \in P$ we fit the function $n \mapsto q(n, d, p)$ with the model:

$$\hat{q} : n \mapsto \frac{\alpha(d, p)}{\sqrt{n}}.$$

For all these fits we compute the maximum error made by the model relatively to the empirical results

$$\varepsilon_{\infty}^{(q)}(d, p) = \max_{n \in N} \left| \frac{\hat{q}(n, d, p)}{q(n, d, p)} - 1 \right|$$

The results of the fits are presented in table A.1. For clarity reasons, we do not present the errors $\varepsilon_{\infty}(d, p)$ for all tuple (d, p) but we show the worst case in each dimension d. They show that even in the worst case, the relative error is quite low. It means that the dependence on n of q can be well modeled by a function decreasing in $\frac{1}{\sqrt{n}}$.

d	1	2	3	4	5	6	7	8
$\max_{p \in P} \varepsilon_{\infty}^{(q)}(d, p)$ (%)	0.249	0.392	0.547	0.574	0.522	0.601	0.565	0.662

Table A.1: *Error made by the model*

The relevance of this model is highlighted by the figure A.1b which compares the simulation results with the model. Undoubtedly, we notice that the model fits well the simulation.

Dependency on the p−value p. Now we assume that the quantile function has the following form $q(n,d,p) = \frac{\alpha(d,p)}{\sqrt{n}}$ and we are looking for the dependence on p. Henceforth we only analyze the behavior of $\alpha(d,p)$. The values of α according to d and p have been retrieved from the previous fitting.

On the figure A.2a we observe the behavior of alpha according to p. We can see the overall shape of the curves does not depend on the dimension: between 0 and about 0.6 the curves are linear and then they increase quickly close to 1. For these reasons we may assume that $\alpha(d,p)$ has the following form:

$$\hat{\alpha} : (d,p) \mapsto \beta(d)\left(p - \gamma(d)\log(1-p)\right)$$

To assess this model we compute its relative error through:

$$\varepsilon_{\infty}^{(\alpha)}(d) = \max_{p\in P}\left|\frac{\hat{\alpha}(d,p)}{\alpha(d,p)} - 1\right|.$$

The results are presented in table A.2. The errors are low but higher than in the first fit. This gap is precisely due to the error propagation by the latter.

d	1	2	3	4	5	6	7	8
$\varepsilon_{\infty}^{(\alpha)}(d)$ (%)	3.682	2.837	2.164	2.432	3.360	3.856	3.757	4.537

Table A.2: *Relative error made by the model*

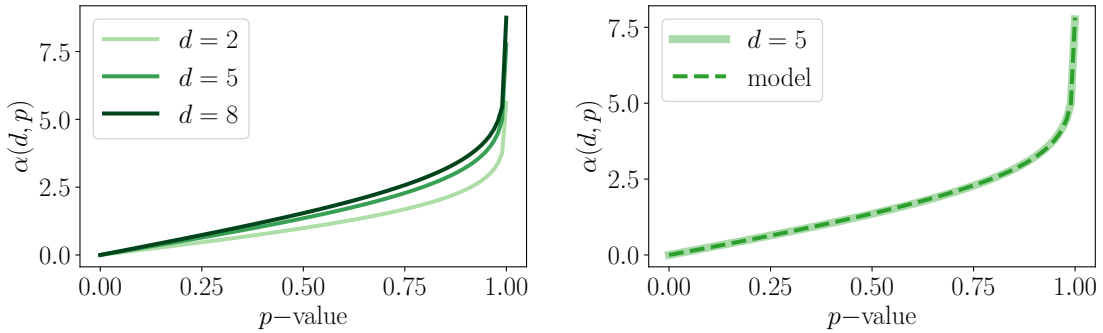
As in the initial fit, we may plot both curves for a given example. The figure A.2b shows the similarity between the simulations and our model for $d = 5$.

Dependency on the dimension d. Finally we have to deals with the dimension. We know that because of the curse of the dimensionality, the high dimensional sampling is hard and the simulation results become less and less relevant.

Through the previous fit, we have to fit two functions β and γ. Let us have a look on their values (table A.3). First we notice that γ does not seem to depend on d, so we assume it as constant. The behavior of β is not as simple.

d	1	2	3	4	5	6	7	8
$\beta(d)$	0.986	1.296	1.480	1.616	1.736	1.827	1.909	1.989
$\gamma(d)$	0.395	0.406	0.416	0.418	0.412	0.409	0.403	0.394

Table A.3: *Fitted values of β and γ according to the dimension d*

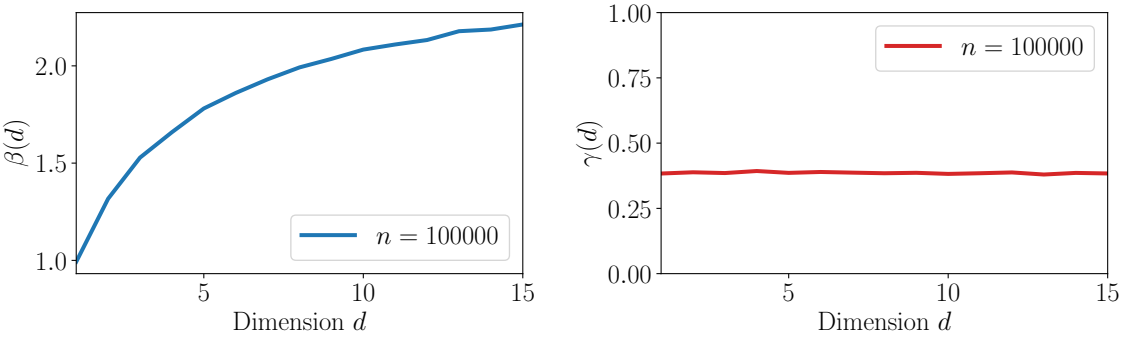


(a) *Influence of the $p-$value p on α for three dimensions d*

(b) *Comparison between the simulation and the model for $d = 5$*

Figure A.2: *Dependency on the $p-$value p*

Fitting with only 8 values is a bit insufficient, therefore we widened the set D to $\{1, 2 \dots 14\}$. We made the same simulations as detailed previously but only with $n = 100000$ to fill the space a bit more. The figure A.3 shows the coefficients β and γ according to the dimension d for $n = 100000$.



(a) *β coefficient according to the dimension*

(b) *γ coefficient according to the dimension*

Figure A.3: *Fitted values of β and γ according to the dimension d ($n = 100000$)*

The plot of γ confirms our assumption. About β, its shape lead us to model it through a logarithmic function

$$\hat{\beta} : d \mapsto \delta \log(d) + \eta.$$

This expression will be incorporated in the final model.

Final model When we combine the influences of all the parameters we may assume the following model for the quantile q:

$$\hat{q} : (n, d, p) \mapsto \frac{a\,(p - b\log(1-p))\,(\log(d) + c)}{\sqrt{n}}, \tag{A.1}$$

where a, b and c are three parameters to fit. Through all our simulation results (about 20000 (n, d, p) configurations) we get

$$a = 0.479, b = 0.407 \text{ and } c = 2.029.$$

The empirical distribution function of the absolute error $|q - \hat{q}(n, d, p)|$ is shown on figure A.4. In a nutshell, this error is very low: for 99% of the simulations, it is lower than 0.005. Thus the model (A.1) is faithful to our simulations and then relevant to compute quantiles.

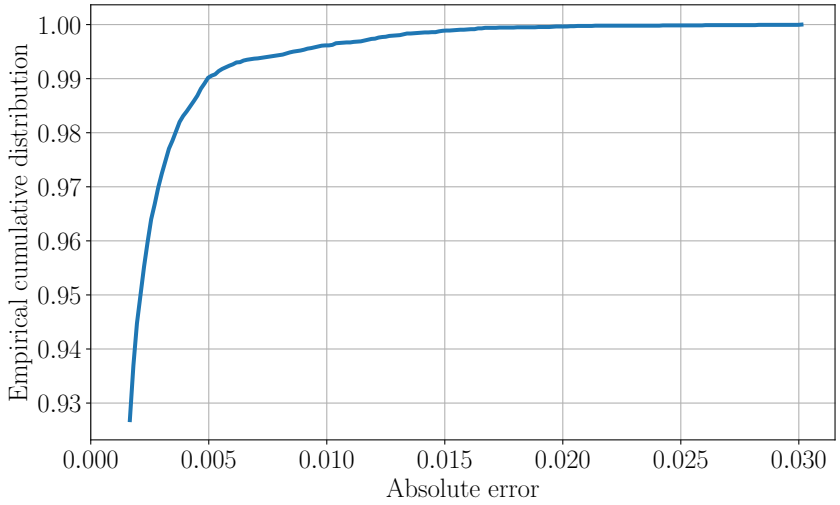


Figure A.4: *Distribution of the absolute error* $|q - \hat{q}|$

B

HFTU materials

LEMMA B.1 *Let X be a random vector of $\mathbb{R}^d$ such that $\mathbb{E}(X)$ exists. Let $u \in \mathbb{R}^d$, then we have the following second order expansion:*

$$\mathbb{E}\left|1 + \alpha \cdot u^T X\right| \underset{\alpha \to 0}{=} 1 + \alpha \cdot u^T \mathbb{E}(X) + o(\alpha^2)$$

Proof. Despite the obviousness of the result, its proof is not as easy as it seems. Actually, we manipulate random vectors and integral operators (expected value) on non-derivable function (absolute value). First we need a smooth approximation of $x \mapsto |x|$. Let $\epsilon > 0$, in particular for all $x \in \mathbb{R}$ we have the following inequalities

$$\frac{x^2}{\sqrt{x^2 + \epsilon^2}} \leq |x| \leq \sqrt{x^2 + \epsilon^2}.$$

Thus by setting $x = 1 + \alpha \cdot u^T X$ and taking the expected value we get:

$$\mathbb{E}\left(\frac{\left(1 + \alpha \cdot u^T X\right)^2}{\sqrt{\left(1 + \alpha \cdot u^T X\right)^2 + \epsilon^2}}\right) \leq \mathbb{E}\left|1 + \alpha \cdot u^T X\right| \leq \mathbb{E}\sqrt{\left(1 + \alpha \cdot u^T X\right)^2 + \epsilon^2}.$$

The goal is to expand bounding functions when $\alpha \to 0$. Let us take the lower bounds

$$g_\epsilon : \alpha \mapsto \int_{\mathbb{R}^d} \frac{\left(1+\alpha\cdot u^T x\right)^2}{\sqrt{\left(1+\alpha\cdot u^T x\right)^2+\epsilon^2}} f(x)\ \mathrm{d}x = \int_{\mathbb{R}^d} G_\epsilon(\alpha,x) f(x)\ \mathrm{d}x$$

We are going to show that g_ϵ is derivable so as to give its Taylor series. In particular we need to derive under the integral operator. For all $x \in \mathbb{R}^d$, the function $\alpha \mapsto G_\epsilon(\alpha, x) f(x)$ is clearly derivable and we have

$$\frac{\partial G_\epsilon}{\partial \alpha}(\alpha,x) = \frac{u^T x\left(1+\alpha\cdot u^T x\right)\left(\left(1+\alpha\cdot u^T x\right)^2+2\epsilon^2\right)}{\left(\left(1+\alpha\cdot u^T x\right)^2+\epsilon^2\right)^{\frac{3}{2}}}$$

With the following inequality, we show that we can derive under the integral operator.

$$\forall \alpha \in \mathbb{R}, \left|\frac{\partial G_\epsilon}{\partial \alpha}(\alpha,x)\right| \leq 2u^T x$$

Thus we have

$$g'_\epsilon(0) = \int_{\mathbb{R}^d} \frac{\partial G_\epsilon}{\partial \alpha}(0,x) f(x)\ \mathrm{d}x = \frac{1+2\epsilon^2}{(1+\epsilon^2)^{\frac{3}{2}}}\,\mathbb{E}(u^T X)$$

We can also show that the function is twice derivable and that

$$g''_\epsilon(0) = -\frac{\epsilon^2(1-2\epsilon^2)}{(1+\epsilon^2)^{\frac{5}{2}}}\,\mathbb{E}\left(\left(u^T X\right)^2\right)$$

Thus the Taylor expansion of the left bound is:

$$g_\epsilon(\alpha) \underset{\alpha\to 0}{=} \frac{1}{\sqrt{1+\epsilon^2}} + \alpha\cdot\frac{1+2\epsilon^2}{(1+\epsilon^2)^{\frac{3}{2}}}\,\mathbb{E}(u^T X) - \alpha^2\frac{\epsilon^2(1-2\epsilon^2)}{2\,(1+\epsilon^2)^{\frac{5}{2}}}\,\mathbb{E}\left(\left(u^T X\right)^2\right) + o(\alpha^2)$$

The same work can be done on the right term, introducing h_ϵ:

$$h_\epsilon : \alpha \mapsto \int_{\mathbb{R}^d} \sqrt{\left(1+\alpha\cdot u^T x\right)^2+\epsilon^2}\cdot f(x)\ \mathrm{d}x = \int_{\mathbb{R}^d} H_\epsilon(\alpha,x) f(x)\ \mathrm{d}x$$

The function H_ϵ is derivable and we have

$$\frac{\partial H_\epsilon}{\partial \alpha}(\alpha,x) = \frac{u^T x\left(1+\alpha\cdot u^T x\right)}{\sqrt{\left(1+\alpha\cdot u^T x\right)^2+\epsilon^2}}$$

We can bound the derivative by $u^T x$ making the derivation under the integral possible. It leads to

$$h'_\epsilon(0) = \int_{\mathbb{R}^d} \frac{\partial H_\epsilon}{\partial \alpha}(0,x) f(x)\ \mathrm{d}x = \frac{1}{\sqrt{1+\epsilon^2}}\,\mathbb{E}(u^T X)$$

The second derivation step is also possible and we get

$$h''_\epsilon(0) = \frac{\epsilon^2}{(1+\epsilon^2)^{\frac{3}{2}}}\,\mathbb{E}\left(\left(u^T X\right)^2\right)$$

Finally, the Taylor expansion of the upper bound is:

$$h_\epsilon(\alpha) \underset{\alpha\to 0}{=} \sqrt{1+\epsilon^2} + \alpha\cdot\frac{1}{\sqrt{1+\epsilon^2}}\,\mathbb{E}(u^T X) - \alpha^2\frac{\epsilon^2}{2\,(1+\epsilon^2)^{\frac{3}{2}}}\,\mathbb{E}\left(\left(u^T X\right)^2\right) + o(\alpha^2)$$

For α small enough, and for all $\epsilon > 0$ we have

$$\begin{aligned}
&\frac{1}{\sqrt{1+\epsilon^2}} + \alpha\cdot\frac{1+2\epsilon^2}{(1+\epsilon^2)^{\frac{3}{2}}}\,\mathbb{E}(u^T X) - \alpha^2\frac{\epsilon^2(1-2\epsilon^2)}{2\,(1+\epsilon^2)^{\frac{5}{2}}}\,\mathbb{E}\left(\left(u^T X\right)^2\right) + l(\alpha)\\
&\qquad\le \mathbb{E}\left|1+\alpha\cdot u^T X\right|\\
&\qquad\le \sqrt{1+\epsilon^2} + \alpha\cdot\frac{1}{\sqrt{1+\epsilon^2}}\,\mathbb{E}(u^T X) - \alpha^2\frac{\epsilon^2}{2\,(1+\epsilon^2)^{\frac{3}{2}}}\,\mathbb{E}\left(\left(u^T X\right)^2\right) + r(\alpha),
\end{aligned}$$

with $r(\alpha)/\alpha^2 \to 0$ and $l(\alpha)/\alpha^2 \to 0$ when $\alpha \to 0$. We can take the limit case when $\epsilon \to 0$, leading to

$$1 + \alpha\,\mathbb{E}(u^T X) + l(\alpha) \le \mathbb{E}\left|1+\alpha\cdot u^T X\right| \le 1 + \alpha\,\mathbb{E}(u^T X) + r(\alpha)$$

Then it means that $\left(\mathbb{E}\left|1+\alpha\cdot u^T X\right| - (1+\alpha\,\mathbb{E}(u^T X))\right)/\alpha^2$ tends to 0 when $\alpha \to 0$, so the expansion of $\mathbb{E}\left|1+\alpha\cdot u^T X\right|$ is proven. □

PROPOSITION B.1 *Let $r > 0$. Let U_d be random vector of $\mathbb{R}^d$ uniformly distributed over the $d-$ball of radius r. The minimum of the folded variance $V_{(u,u_0)}(U_d)$ is reached at $u_0 = 0$, and is given by:*

$$V_{\mathcal{H}^*}(U_d) = r^2\cdot\left(\frac{d}{d+2} - \left(\frac{a_d}{d+1}\right)^2\right).$$

Furthermore, the folding ratio ϕ_d of U_d does not depend on r and is given by

$$\phi_d = \phi(U_d) = 1 - \frac{(d+2)a_d^2}{d(d+1)^2} \quad \textit{with} \quad a_d = \frac{2}{\sqrt{\pi}}\frac{\Gamma\left(\frac{d}{2}+1\right)}{\Gamma\left(\frac{d}{2}+\frac{1}{2}\right)} = \begin{cases} \dfrac{4^p}{\pi}\dfrac{p!(p-1)!}{(2p-1)!} & \textit{when } d = 2p \\ \dfrac{(2p+1)!}{4^p(p!)^2} & \textit{when } d = 2p+1 \end{cases}$$

Proof. Let $r > 0$. Let U_d be random vector of $\mathbb{R}^d$ uniformly distributed over the $d-$ball of radius r. Let $u \in \mathcal{S}^d, u_0 \in [-r, r]$. First we show that the folded variance of U_d about (u, u_0) is given by

$$V_{(u,u_0)}(U_d) = \operatorname{Tr}\operatorname{Cov}(U_d) + r^2\cdot G\left(\arcsin(u_0/r)\right)$$

with

$$G \quad : \quad \begin{array}{ccc} \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] & \to & \mathbb{R} \\ \alpha & \mapsto & \sin^2\alpha - a_d^2 \left(\frac{\cos^{d+1}\alpha}{d+1} + \sin\alpha \int_0^\alpha \cos^d\theta \ \mathrm{d}\theta \right)^2 \end{array}$$

with $a_d = \dfrac{2}{\sqrt{\pi}} \dfrac{\Gamma\left(\frac{d}{2}+1\right)}{\Gamma\left(\frac{d}{2}+\frac{1}{2}\right)}$.

The density of U_d is constant within the ball equal to $1/B_d(r)$ where $B_d(r)$ is the volume of the $d-$ball with radius r (it is null elsewhere):

$$B_d(r) = \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}+1\right)} r^d = c_d \cdot r^d$$

For symmetry reasons, the orientation of the best hyperplan has no impact so we can set $u = (-1\ 0 \cdots 0)$. In this configuration,

$$\mathbb{E}(h_\lambda(U_d)) = h_\lambda(\mathbb{E}(U_d)) = h_\lambda(0) = u_0.$$

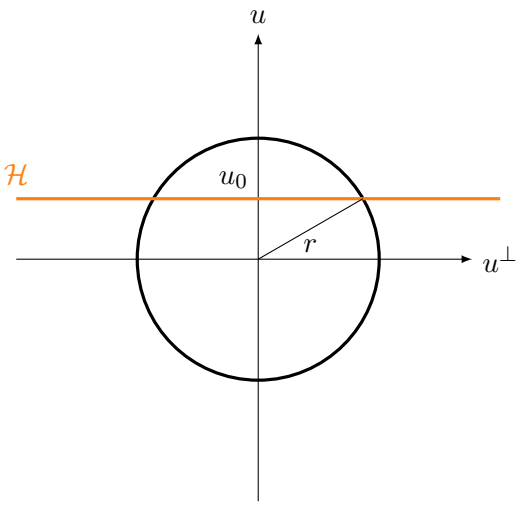


Figure B.1: *Hyperplan $\mathcal{H}$ splitting a $d-$ball of radius r.*

The hard step is the computation of $\mathbb{E}\,|h_\lambda(U_d)|$. The intersection of a $d-$ball with an hyperplan $\mathcal{H}$ is a $(d-1)-$ball. In particular, if u is a unit normal vector of $\mathcal{H}$ and if the point $zu, z \in [-r, r]$ belongs to this intersection, then the radius of the $(d-1)-$ball

is $\sqrt{r^2-z^2}$. The expected value is then

$$\begin{aligned}\mathbb{E}\,|h_\lambda(U_d)| &= \int_{-r}^{r} |z-u_0|\cdot B_{d-1}\left(\sqrt{r^2-z^2}\right)\cdot\frac{1}{B_d(r)}\,\mathrm{d}z\\ &= \frac{c_{d-1}}{c_d}\cdot\frac{1}{r^d}\cdot\left(\int_{-r}^{u_0}(u_0-z)\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\right.\\ &\qquad\left.+\int_{u_0}^{r}(z-u_0)\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\right)\end{aligned}$$

The integrals are quite similar. Changing $z\mapsto -z$ on the first one, we get

$$\int_{-r}^{u_0}(u_0-z)\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z=\int_{-u_0}^{r}(z+u_0)\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z.$$

The expected value can then be written as

$$\begin{aligned}\mathbb{E}\,|h_\lambda(U_d)| = \frac{c_{d-1}}{c_d}\cdot\frac{1}{r^d}\cdot\left(\int_{-u_0}^{r} z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z+\int_{u_0}^{r} z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\right.\\ \left.+u_0\left(\int_{-u_0}^{r}\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z-\int_{u_0}^{r}\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\right)\right)\end{aligned}$$

Let us consider the two first terms in the bracket. They can be simplified as the integrated function is odd and a primitive is obvious.

$$\begin{aligned}&\int_{-u_0}^{r} z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z+\int_{u_0}^{r} z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\\ &\qquad=\int_{-u_0}^{u_0} z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z+2\int_{u_0}^{r} z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\\ &\qquad=\int_{u_0}^{r} 2z\cdot\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\\ &\qquad=\left[-\frac{2}{d+1}\left(r^2-z^2\right)^{\frac{d+1}{2}}\right]_{u_0}^{r}\\ &\qquad=\frac{2}{d+1}\left(r^2-u_0^2\right)^{\frac{d+1}{2}}\\ &\qquad=\frac{2r^{d+1}}{d+1}\left(1-\left(\frac{u_0}{r}\right)^2\right)^{\frac{d+1}{2}}\end{aligned}$$

The two others can also be simplified.

$$\begin{aligned}\int_{-u_0}^{r}\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z-\int_{u_0}^{r}\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z&=\int_{-u_0}^{u_0}\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\\ &=2\int_{0}^{u_0}\left(r^2-z^2\right)^{\frac{d-1}{2}}\,\mathrm{d}z\\ &=2r^d\int_{0}^{\frac{u_0}{r}}\left(1-x^2\right)^{\frac{d-1}{2}}\,\mathrm{d}x\end{aligned}$$

Injecting in the expected value, we get,

$$\begin{aligned}\mathbb{E}\,|h_\lambda(U_d)| &= \frac{c_{d-1}}{c_d}\cdot\frac{2}{r^d}\cdot\left(\frac{r^{d+1}}{d+1}\left(1-\left(\frac{u_0}{r}\right)^2\right)^{\frac{d+1}{2}}+u_0r^d\int_0^{\frac{u_0}{r}}\left(1-x^2\right)^{\frac{d-1}{2}}\ \mathrm{d}x\right)\\ &= 2r\cdot\frac{c_{d-1}}{c_d}\left(\frac{1}{d+1}\left(1-\left(\frac{u_0}{r}\right)^2\right)^{\frac{d+1}{2}}+\frac{u_0}{r}\int_0^{\frac{u_0}{r}}\left(1-x^2\right)^{\frac{d-1}{2}}\ \mathrm{d}x\right)\end{aligned}$$

Let us write the whole expression to minimize

$$\begin{aligned}&\mathbb{E}\left(h_\lambda(U_d)\right)^2-\mathbb{E}\left(|h_\lambda(U_d)|\right)^2\\ &= u_0^2-4\left(\frac{c_{d-1}}{c_d}\right)^2r^2\left(\frac{1}{d+1}\left(1-\left(\frac{u_0}{r}\right)^2\right)^{\frac{d+1}{2}}+\frac{u_0}{r}\int_0^{\frac{u_0}{r}}\left(1-x^2\right)^{\frac{d-1}{2}}\ \mathrm{d}x\right)^2\\ &= r^2\left(\left(\frac{u_0}{r}\right)^2-4\left(\frac{c_{d-1}}{c_d}\right)^2\left(\frac{1}{d+1}\left(1-\left(\frac{u_0}{r}\right)^2\right)^{\frac{d+1}{2}}+\frac{u_0}{r}\int_0^{\frac{u_0}{r}}\left(1-x^2\right)^{\frac{d-1}{2}}\ \mathrm{d}x\right)^2\right)\end{aligned}$$

We notice that the bracket only depends on the ratio u_0/r. However, for a practical point of view it is better to use the variable $\sin\alpha = u_0/r$. As $u_0/r \in [-1,1]$, we have $\alpha \in [-\pi/2, \pi/2]$. The key function is then

$$G:\alpha\mapsto\sin^2\alpha-a_d^2\left(\frac{\cos^{d+1}\alpha}{d+1}+\sin\alpha\int_0^{\sin\alpha}\left(1-x^2\right)^{\frac{d-1}{2}}\ \mathrm{d}x\right)^2$$

The integral can also be re-parameterized by setting $x=\sin\theta$. It gives

$$\int_0^{\sin\alpha}\left(1-x^2\right)^{\frac{d-1}{2}}\ \mathrm{d}x=\int_0^{\alpha}\cos^d\theta\ \mathrm{d}\theta.$$

Finally,

$$G(\alpha)=\sin^2\alpha-a_d^2\left(\frac{\cos^{d+1}\alpha}{d+1}+\sin\alpha\int_0^{\alpha}\cos^d\theta\ \mathrm{d}\theta\right)^2$$

Now, we only need to study the function G, first we can easily check that this function is even, so we will focus on the case $\alpha\geq 0$. In all cases, it can naturally be factorized:

$$\begin{aligned}G(\alpha)=&\left(\sin\alpha-a_d\left(\frac{\cos^{d+1}\alpha}{d+1}+\sin\alpha\int_0^{\alpha}\cos^d\theta\ \mathrm{d}\theta\right)\right)\\ &\cdot\left(\sin\alpha+a_d\left(\frac{\cos^{d+1}\alpha}{d+1}+\sin\alpha\int_0^{\alpha}\cos^d\theta\ \mathrm{d}\theta\right)\right)\end{aligned}$$

We also recall that G is a negative function (the folded variance if lower than $\operatorname{Tr}\Sigma$). As the second bracket is positive, we have for $\alpha \geq 0$

$$\sin\alpha \leq a_d \left(\frac{\cos^{d+1}\alpha}{d+1} + \sin\alpha \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right).$$

Let us compute its first derivative of G:

$$\begin{aligned} G'(\alpha) &= 2\sin(\alpha)\cos(\alpha) \\ &\quad - 2a_d^2 \left(\frac{\cos^{d+1}\alpha}{d+1} + \sin\alpha \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right) \\ &\quad \times \left(-\cos^d(\alpha)\sin\alpha + \sin\alpha\cos^d(\alpha) + \cos\alpha \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right) \\ &= 2\cos\alpha \left(\sin\alpha - a_d^2 \left(\frac{\cos^{d+1}\alpha}{d+1} + \sin\alpha \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right) \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right) \end{aligned}$$

With the bounds on $\sin\alpha$, we have also a bound for $G'(\alpha)$

$$\begin{aligned} G'(\alpha) &\geq 2\cos\alpha \left(\sin\alpha - \sin\alpha \cdot a_d \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right) \\ &\geq 2\cos\alpha\sin\alpha \left(1 - a_d \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta \right) \end{aligned}$$

However,

$$a_d \int_0^\alpha \cos^d\theta \, \mathrm{d}\theta = \frac{1}{c_d} \int_{-\alpha}^\alpha c_{d-1}\cos^d\theta \, \mathrm{d}\theta \leq \frac{1}{c_d} \cdot c_d = 1$$

Thus G' is positive when $\alpha \geq 0$. So G is an increasing function on $[0, \pi/2]$. But as G is even, it decreases on $[-\pi/2, 0]$. Finally it reaches its minimum when $\alpha = 0$ and the case $\alpha = 0$ corresponds to $u_0 = 0$. □

References

[1] Ahmad, S., Lavin, A., Purdy, S., and Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing 262* (2017), 134–147.

[2] Amin, A. A., and Ibne, R. M. B. A novel svm-knn-pso ensemble method for intrusion detection system. *Applied Soft Computing 38* (2016), 360–372.

[3] Amor, N. B., Benferhat, S., and Elouedi, Z. Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing* (2004), ACM, pp. 420–424.

[4] Anderson, T. W., Darling, D. A., et al. Asymptotic theory of certain" goodness of fit" criteria based on stochastic processes. *The annals of mathematical statistics 23*, 2 (1952), 193–212.

[5] Anscombe, F. J. Rejection of outliers. *Technometrics 2*, 2 (1960), 123–146.

[6] Balkema, A. A., and De Haan, L. Residual life time at great age. *The Annals of probability* (1974), 792–804.

[7] Barichievy, C., Angeler, D. G., Eason, T., Garmestani, A. S., Nash, K. L., Stow, C. A., Sundstrom, S., and Allen, C. R. A method to detect discontinuities in census data. *Ecology and Evolution 8*, 19 (2018), 9614–9623.

[8] Beirlant, J., Goegebeur, Y., Segers, J., and Teugels, J. L. *Statistics of extremes: theory and applications*. John Wiley & Sons, 2006.

[9] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials 16*, 1 (2013), 303–336.

[10] Bivens, A., Palagiri, C., Smith, R., Szymanski, B., Embrechts, M., et al. Network-based intrusion detection using neural networks. *Intelligent Engineering Systems through Artificial Neural Networks 12*, 1 (2002), 579–584.

[11] Brent, R. P. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[12] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: identifying density-based local outliers. In *ACM sigmod record* (2000), vol. 29, ACM, pp. 93–104.

[13] Buczak, A. L., and Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials 18*, 2 (2015), 1153–1176.

[14] Casas, P., Mazel, J., and Owezarski, P. Unada: Unsupervised network anomaly detection using sub-space outliers ranking. In *International Conference on Research in Networking* (2011), Springer, pp. 40–51.

[15] Casas, P., Mazel, J., and Owezarski, P. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications 35*, 7 (2012), 772–783.

[16] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 15.

[17] Cheng, M.-Y., and Hall, P. Calibrating the excess mass and dip tests of modality. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 60*, 3 (1998), 579–589.

[18] Clay, P. R. Shortest connection networks and some generalizations. *The Bell System Technical Journal 36*, 6 (1957), 1389–1401.

[19] Coles, S., Bawa, J., Trenner, L., and Dorazio, P. *An introduction to statistical modeling of extreme values*, vol. 208. Springer, 2001.

[20] Comaniciu, D., and Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 5 (2002), 603–619.

[21] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[22] Cummings, B. B., Marshall, J. L., Tukiainen, T., Lek, M., Donkervoort, S., Foley, A. R., Bolduc, V., Waddell, L. B., Sandaradura, S. A., O'Grady, G. L., et al. Improving genetic diagnosis in mendelian disease with transcriptome sequencing. *Science translational medicine 9*, 386 (2017), eaal5209.

[23] Denning, D. E. An intrusion-detection model. *IEEE Transactions on software engineering*, 2 (1987), 222–232.

[24] Dharmadhikari, S., and Joag-Dev, K. *Unimodality, convexity, and applications*. Elsevier, 1988.

[25] Ding, K., Li, J., Bhanushali, R., and Liu, H. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining* (2019), SIAM, pp. 594–602.

[26] Dromard, J., Roudière, G., and Owezarski, P. Unsupervised network anomaly detection in real-time on big data. In *East European Conference on Advances in Databases and Information Systems* (2015), Springer, pp. 197–206.

[27] Dromard, J., Roudière, G., and Owezarski, P. Online and scalable unsupervised network anomaly detection method. *IEEE Transactions on Network and Service Management 14*, 1 (2016), 34–47.

[28] E, M. M. Some continuous monte carlo methods for the dirichlet problem. *The Annals of Mathematical Statistics* (1956), 569–589.

[29] Efron, B. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*. Springer, 1992, pp. 569–593.

[30] Eskin, E. Anomaly detection over noisy data using learned probability distributions.

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

[32] Feng, Y., and Hamerly, G. Pg-means: learning the number of clusters in data. In *Advances in neural information processing systems* (2007), pp. 393–400.

[33] Ferragut, E. M., Laska, J., and Bridges, R. A. A new, principled approach to anomaly detection. In *2012 11th International Conference on Machine Learning and Applications* (2012), vol. 2, IEEE, pp. 210–215.

[34] Figueiredo, M. A. T., and Jain, A. K. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 3 (2002), 381–396.

[35] Fisher, R. A., and Tippett, L. H. C. Limiting forms of the frequency distribution of the largest or smallest member of a sample. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1928), vol. 24, Cambridge University Press, pp. 180–190.

[36] Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10* (2010).

[37] Frey, B. J., and Dueck, D. Clustering by passing messages between data points. *science 315*, 5814 (2007), 972–976.

[38] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security 28*, 1-2 (2009), 18–28.

[39] Gebhardt, J., Goldstein, M., Shafait, F., and Dengel, A. Document authentication using printing technique features and unsupervised anomaly detection. In *2013 12th International Conference on Document Analysis and Recognition* (2013), IEEE, pp. 479–483.

[40] Gnedenko, B. Sur la distribution limite du terme maximum d'une serie aleatoire. *Annals of mathematics* (1943), 423–453.

[41] Golan, I., and El-Yaniv, R. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems* (2018), pp. 9758–9769.

[42] Goldstein, M., and Dengel, A. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track* (2012), 59–63.

[43] Greenwood, J. A., Landwehr, J. M., Matalas, N. C., and Wallis, J. R. Probability weighted moments: definition and relation to parameters of several distributions expressable in inverse form. *Water resources research 15*, 5 (1979), 1049–1054.

[44] Grimshaw, S. D. Computing maximum likelihood estimates for the generalized pareto distribution. *Technometrics 35*, 2 (1993), 185–191.

[45] Grubbs, F. E. Procedures for detecting outlying observations in samples. *Technometrics 11*, 1 (1969), 1–21.

[46] Guilherme, F., and Philippe, O. Automated classification of network traffic anomalies. In *ICSPCS* (2009).

[47] Guillot, A., Fontugne, R., Winter, P., Merindol, P., King, A., Dainotti, A., and Pelsser, C. Chocolatine: Outage detection for internet background radiation. *arXiv preprint arXiv:1906.04426* (2019).

[48] Hall, P., and York, M. On the calibration of silverman's test for multimodality. *Statistica Sinica* (2001), 515–536.

[49] Hämäläinen, W., and Webb, G. I. A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery 33*, 2 (2019), 325–377.

[50] Hamerly, G., and Elkan, C. Learning the k in k-means. In *Advances in neural information processing systems* (2004), pp. 281–288.

[51] Hansen, M. H., and Yu, B. Model selection and the principle of minimum description length. *Journal of the American Statistical Association 96*, 454 (2001), 746–774.

[52] Hartigan, J., and Mohanty, S. The runt test for multimodality. *Journal of Classification 9*, 1 (1992), 63–70.

[53] Hartigan, J. A., and Hartigan, P. M. The dip test of unimodality. *The annals of Statistics 13*, 1 (1985), 70–84.

[54] Hochenbaum, J., Vallis, O. S., and Kejariwal, A. Automatic anomaly detection in the cloud via statistical learning. *arXiv preprint arXiv:1704.07706* (2017).

[55] Hosking, J. R., and Wallis, J. R. Parameter and quantile estimation for the generalized pareto distribution. *Technometrics 29*, 3 (1987), 339–349.

[56] Houerbi, K. R., Salamatian, K., and Kamoun, F. Scan surveillance in internet networks. In *International Conference on Research in Networking* (2009), Springer, pp. 614–625.

[57] Hu, W., Liao, Y., and Vemuri, V. R. Robust support vector machines for anomaly detection in computer security. In *ICMLA* (2003), pp. 168–174.

[58] Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), ACM, pp. 387–395.

[59] Jain, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters 31*, 8 (2010), 651–666.

[60] Jiong, Z., and Mohammad, Z. A hybrid network intrusion detection technique using random forests. In *Availability Reliability and Security 2006. ARES 2006. The First International Conference on* (2006), IEEE, pp. 8–pp.

[61] Joshi, S. S., and Phoha, V. V. Investigating hidden markov models capabilities in anomaly detection. In *Proceedings of the 43rd annual Southeast regional conference-Volume 1* (2005), ACM, pp. 98–103.

[62] Kalogeratos, A., and Likas, A. Dip-means: an incremental clustering method for estimating the number of clusters. In *Advances in neural information processing systems* (2012), pp. 2393–2401.

[63] Karp, R. M., Shenker, S., and Papadimitriou, C. H. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS) 28*, 1 (2003), 51–55.

[64] Khintchine, A. On unimodal distributions. *Izvestiya Nauchno-Issledovatel'skogo Instituta Matematiki i Mekhaniki 2*, 2 (1938), 1–7.

[65] Kim, Y., Lau, W. C., Chuah, M. C., and Chao, H. J. Packetscore: Statistics-based overload control against distributed denial-of-service attacks. In *IEEE INFOCOM 2004* (2004), vol. 4, IEEE, pp. 2594–2604.

[66] Kind, A., Stoecklin, M. P., and Dimitropoulos, X. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management 6*, 2 (2009), 110–121.

[67] Knyazev, A. V. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing 23*, 2 (2001), 517–541.

[68] Komiyama, J., Ishihata, M., Arimura, H., Nishibayashi, T., and Minato, S.-i. Statistical emerging pattern mining with multiple testing correction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), ACM, pp. 897–906.

[69] Kraft, D. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (1988).

[70] Krishna, V. B., Iyer, R. K., and Sanders, W. H. Arima-based modeling and validation of consumption readings in power grids. In *International Conference on Critical Information Infrastructures Security* (2015), Springer, pp. 199–210.

[71] Kurihara, K., and Welling, M. Bayesian k-means as a "maximization-expectation" algorithm. *Neural computation 21*, 4 (2009), 1145–1172.

[72] Laftah, A.-Y. W., Ali, O. Z., and Ahmad, N. M. Z. Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. *Expert Systems with Applications 67* (2017), 296–303.

[73] Lakhina, A., Crovella, M., and Diot, C. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review* (2005), vol. 35, ACM, pp. 217–228.

[74] Lange, T., Roth, V., Braun, M. L., and Buhmann, J. M. Stability-based validation of clustering solutions. *Neural computation 16*, 6 (2004), 1299–1323.

[75] Laurikkala, J., Juhola, M., Kentala, E., Lavrac, N., Miksch, S., and Kavsek, B. Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology* (2000), vol. 1, pp. 20–24.

[76] Li, W., Mahadevan, V., and Vasconcelos, N. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence 36*, 1 (2013), 18–32.

[77] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. The 1999 darpa off-line intrusion detection evaluation. *Computer networks 34*, 4 (2000), 579–595.

[78] Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschogrod, D., Cunningham, R. K., et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00* (2000), vol. 2, IEEE, pp. 12–26.

[79] Lleti, R., Ortiz, M. C., Sarabia, L. A., and Sánchez, M. S. Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes. *Analytica Chimica Acta 515*, 1 (2004), 87–100.

[80] Mackay, E. B., Challenor, P. G., and Bahaj, A. S. A comparison of estimators for the generalised pareto distribution. *Ocean Engineering 38*, 11-12 (2011), 1338–1346.

[81] Mahoney, M. V., and Chan, P. K. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *International Workshop on Recent Advances in Intrusion Detection* (2003), Springer, pp. 220–237.

[82] Mammen, E., Marron, J. S., and Fisher, N. I. Some asymptotics for multimodality tests based on kernel density estimates. *Probability Theory and Related Fields 91*, 1 (1992), 115–132.

[83] Manzoor, E., Lamba, H., and Akoglu, L. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), ACM, pp. 1963–1972.

[84] Maurus, S., and Plant, C. Skinny-dip: Clustering in a sea of noise. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (2016), ACM, pp. 1055–1064.

[85] Maurus, S., and Plant, C. Let's see your digits: Anomalous-state detection using benford's law. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), ACM, pp. 977–986.

[86] Mazel, J., Fontugne, R., and Fukuda, K. A taxonomy of anomalies in backbone network traffic. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)* (2014), IEEE, pp. 30–36.

[87] McHugh, J. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC) 3*, 4 (2000), 262–294.

[88] Minnotte, M. C. *A test of mode existence with applications to multimodality.* PhD thesis, Rice University, 1993.

[89] Minnotte, M. C. Nonparametric testing of the existence of modes. *The Annals of Statistics* (1997), 1646–1660.

[90] Mirsky, Y., Doitshman, T., Elovici, Y., and Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. *NDSS 5* (2018), 2.

[91] Mukherjee, B., Heberlein, L. T., and Levitt, K. N. Network intrusion detection. *IEEE network 8*, 3 (1994), 26–41.

[92] Müller, D. W., and Sawitzki, G. Excess mass estimates and tests for multimodality. *Journal of the American Statistical Association 86*, 415 (1991), 738–746.

[93] Muller, M. E. A note on a method for generating points uniformly on n-dimensional spheres. *Communications of the ACM 2*, 4 (1959), 19–20.

[94] Nicolau, M., McDermott, J., et al. A hybrid autoencoder and density estimation model for anomaly detection. In *International Conference on Parallel Problem Solving from Nature* (2016), Springer, pp. 717–726.

[95] Pang, G., Cao, L., Chen, L., and Liu, H. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), ACM, pp. 2041–2050.

[96] Papadimitriou, S., Kitagawa, H., Gibbons, P. B., and Faloutsos, C. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)* (2003), IEEE, pp. 315–326.

[97] Patcha, A., and Park, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks 51*, 12 (2007), 3448–3470.

[98] Pelleg, D., Moore, A. W., et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml* (2000), vol. 1, pp. 727–734.

[99] Pickands III, J., et al. Statistical inference using extreme order statistics. *the Annals of Statistics 3*, 1 (1975), 119–131.

[100] Portnoy, L. *Intrusion detection with unlabeled data using clustering*. PhD thesis, Columbia University, 2000.

[101] Powell, M. J. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in optimization and numerical analysis*. Springer, 1994, pp. 51–67.

[102] Qin, Y., Lyu, J., and Lirui, J. An improved arima-based traffic anomaly detection algorithm for wireless sensor networks. *International Journal of Distributed Sensor Networks 12*, 1 (2016), 9653230.

[103] Rosner, B. Percentage points for a generalized esd many-outlier procedure. *Technometrics 25*, 2 (1983), 165–172.

[104] Rousseeuw, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics 20* (1987), 53–65.

[105] Rousseeuw, P. J., and Leroy, A. M. *Robust regression and outlier detection*, vol. 589. John wiley & sons, 2005.

[106] Rozál, G. P. M., and Hartigan, J. The map test for multimodality. *Journal of Classification 11*, 1 (1994), 5–36.

[107] Ryan, D., Denman, S., Fookes, C., and Sridharan, S. Textures of optical flow for real-time anomaly detection in crowds. In *2011 8th IEEE international conference on advanced video and signal based surveillance (AVSS)* (2011), IEEE, pp. 230–235.

[108] Salem, O., Liu, Y., Mehaoua, A., and Boutaba, R. Online anomaly detection in wireless body area networks for reliable healthcare monitoring. *IEEE journal of biomedical and health informatics 18*, 5 (2014), 1541–1551.

[109] Sawitzki, G. The excess mass approach and the analysis of multi-modality. In *From Data to Knowledge*. Springer, 1996, pp. 203–211.

[110] Schmitt, N., and Westerhoff, F. On the bimodality of the distribution of the s&p 500's distortion: Empirical evidence and theoretical explanations. *Journal of Economic Dynamics and Control 80* (2017), 34–53.

[111] Schubert, E., Zimek, A., and Kriegel, H.-P. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining* (2014), SIAM, pp. 542–550.

[112] Shadi, A., Monther, A., and Bani, Y. M. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science* (2017).

[113] Sherman, J., and Morrison, W. J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics 21*, 1 (1950), 124–127.

[114] Siffer, A. Intelligent thresholding. In *Proceedings of the C&ESAR conference* (2018).

[115] Siffer, A., Fouque, P.-A., Termier, A., and Largouët, C. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), ACM, pp. 1067–1075.

[116] Siffer, A., Fouque, P.-A., Termier, A., and Largouët, C. Are your data gathered? In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), ACM, pp. 2210–2218.

[117] Silverman, B. W. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society: Series B (Methodological) 43*, 1 (1981), 97–99.

[118] Silverman, B. W. Some properties of a test for multimodality based on kernel density estimates. *Probability statistics and analysis 79* (1983), 248–259.

[119] Stoepker, I. Testing for multimodality, 2016.

[120] Stolfo, S., et al. Kdd cup 1999 dataset. uci kdd repository, 1999.

[121] Sugar, C. A., and James, G. M. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association 98*, 463 (2003), 750–763.

[122] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (2009), IEEE, pp. 1–6.

[123] Thomas, A., Clémençon, S., Gramfort, A., and Sabourin, A. Anomaly detection in extreme regions via empirical mv-sets on the sphere. In *AISTATS* (2017), pp. 1011–1019.

[124] Tibshirani, R., Walther, G., and Hastie, T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63*, 2 (2001), 411–423.

[125] Wang, C., and Chen, G. A new hybrid estimation method for the generalized pareto distribution. *Communications in Statistics-Theory and Methods 45*, 14 (2016), 4285–4294.

[126] Wu, S. X., and Banzhaf, W. The use of computational intelligence in intrusion detection systems: A review. *Applied soft computing 10*, 1 (2010), 1–35.

[127] Yao, X., Cafaro, J., McLaughlin, A. J., Postma, F. R., Paul, D. L., Awatramani, G., and Field, G. D. Gap junctions contribute to differential light adaptation across direction-selective retinal ganglion cells. *Neuron 100*, 1 (2018), 216–228.

[128] Yu, W., Cheng, W., Aggarwal, C. C., Zhang, K., Chen, H., and Wang, W. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), ACM, pp. 2672–2681.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Nouvelles méthodes statistiques pour la fouille de données

Mots clés : Méthodes statistiques, Détection d'anomalies, Test d'unimodalité, flux de données

Resumé : Cette thèse propose de nouveaux algorithmes statistiques dans deux domaines différents de la fouille de données: la détection d'anomalies et le test d'unimodalité. Premièrement, une nouvelle méthode non-supervisée permettant de détecter des anomalies dans des flux de données est développée. Celle-ci se base sur le calcul de seuils probabilistes, eux-mêmes utilisés pour discriminer les observations anormales. La force de cette méthode est sa capacité à s'exécuter automatiquement sans connaissance préalable ni hypothèse sur le flux de données d'intérêt. De même, l'aspect générique de l'algorithme lui permet d'opérer dans des domaines d'application variés. En particulier, nous développons un cas d'usage en cyber-sécurité. Cette thèse développe également un nouveau test d'unimodalité qui permet de déterminer si une distribution de données comporte un ou plusieurs modes. Ce test est nouveau par deux aspects: sa capacité à traiter des distributions multivariées mais également sa faible complexité, lui permettant alors d'être appliqué en temps réel sur des flux de données. Cette composante plus fondamentale a principalement des applications dans d'autres domaines du *data mining* tels que le *clustering*. Un nouvel algorithme cherchant incrémentalement le paramétrage de $k-$means est notamment détaillé à la fin de ce manuscrit.

Title : New statistical methods for data mining

Keywords : Statistical methods, Anomaly detection, Unimodality testing, Streaming data

Abstract : This thesis proposes new statistical algorithms in two different data mining areas: anomaly detection and unimodality testing. First, a new unsupervised method for detecting outliers in streaming data is developed. It is based on the computation of probabilistic thresholds, which are themselves used to discriminate against abnormal observations. The strength of this method is its ability to run automatically without prior knowledge or hypothesis about the input data. Similarly, the generic aspect of the algorithm makes it able to operate in various fields. In particular, we develop a cyber-security use case. This thesis also proposes a new unimodality test which determines whether a data distribution has one or several modes. This test is new in two respects: its ability to handle multivariate distributions but also its low complexity, allowing it to be applied on streaming data. This more fundamental component has applications mainly in other areas of data mining such as clustering. A new algorithm incrementally searching for the $k-$means parameter setting is notably detailed at the end of this manuscript.