Analysis on MDL Estimators and A Practical Study on Machine Learning Based Cyber Security

宮本, 耕平

Analysis on MDL Estimators and A Practical Study on Machine Learning Based Cyber Security

Kohei Miyamoto

2024

Department of Informatics,

Graduate School of Information Science and Electrical Engineering,

Kyushu University

Contents

Abstract

Nowadays machine learning techniques, which allow us to make inferences based on collected data, become increasingly important for various domains. Therefore both theoretical and practical research and development on it are also important.

In this thesis, we treat both theoretical and practical topics. As a theoretical topic, we argue on performance of a probability density estimator based on the minimum description length (MDL) principle, called MDL estimator. An MDL estimator is induced by a two part code which is a kind of universal code. An important property of MDL estimators is that a statistical risk of an MDL estimator is bounded by using the redundancy of the two part code which induces the MDL estimator. Therefore a method of constructing a two part code with small redundancy leads to an MDL estimator with high performance. There is a well known method of constructing using the Fisher information matrix of the target class of models. However it have focused on only exponential families, and a construction of two part codes good for non-exponential families has not been known.

We improve the construction of two part code so that the two part code has small redundancy for non-exponential families under some regularity conditions. We also provide performance guarantees for the MDL estimator induced by the improved two part code. Further we give examples of non-exponential families for which our method works well, mixture families and contaminated Gaussian families.

Our construction of two part codes is based on model enlargement by using fiber bundles of local exponential families. In analysis for contaminated Gaussian families, to give an upper bound of statistical risk of our MDL estimators, we also use a technique of using typical sets.

Further we discuss the determinant of Fisher information of Gaussian distributions. This will help us to study on MDL estimatiors for Gaussian mixture models, which is an important example of non-exponential families.

As a practical topic, we argue on an application of machine learning for cyber security. In particular we discuss network intrusion detection system (NIDS) based on machine learning. NIDSs monitor network traffics and detect anomalies or suspicious behavior in it. When an NIDS detects such anomaly or behavior, it issues an alert which provides security operators with information useful for security operation to defend the network.

Since new approaches of cyber attack are emerging one after another, NIDSs should be updated so that they can detect new atacks. However designing detection rules usually requires expert's knowledge and time, and it is hard to update NIDSs quickly. Machine learning approach may ease this difficulty. If we have dataset of captured traffic including activities of cyber attacks, machine learning models may be possible to learn rules to detect the attacks from the data. This approach requires appropriate dataset. However experts do not need to design new their handmade rules and can focus on more critical roles in security operation like detailed investigation of detected malicious activities.

In this thesis, we propose a machine learning based approach for NIDSs which is based on packet-level classification task. In our approach, we use features extracted from each captured packet as input of a model and judge whether each traffic flow is malicious or not based on classification results for each packet in the flow.

Chapter 1

Introduction

The main topic of this thesis is on the density estimation problem. In statistical inference or machine learning, the density estimation is an essential problem, where we estimate a probability density function behind of given data. In this thesis, we argue a density estimator based on the minimum description length (MDL) principle. The MDL principle [1] supposes that minimizing the code length, which we need to encode given data, leads to a good estimate for the distribution behind the data.

We call density estimates based on the MDL principle as MDL estimates. We also call density estimators which map given data to the MDL estimate as MDL estimators. An MDL estimate for given data is defined as a minimizer of a code length of a two part code for the data. We argue performance guarantees of MDL estimators in this thesis.

Another topic of this thesis is an application of machine learning to cyber security. Detecting cyber attacks is crucial to execute appropriate actions for the attacks. Monitoring network traffics is a basic process to detect cyber attacks. However investigating all monitored traffics manually is impossible, because the amount of the traffics is usually too large. Therefore a system which automatically detect suspicious behaviors in the monitored traffics is required. In this thesis, we develop such a system based on machine learning and provide experimental results using a bench mark dataset.

1.1 MDL estimation

First, we provide an introduction the main topic of this thesis on MDL based density estimation.

1.1.1 Motivation

There is an important property of MDL estimators known by the Barron-Cover's theorem [2, 3]. The theorem claims that their losses measured by Rényi divergence [4, 5] have an upper bound given by using redundancy of the two part code which induces them. This fact implies that when we design a two part code whose redundancy is small, we obtain a good MDL estimator. This provides a justification of the MDL principle.

Then, we should design a two part code whose redundancy is as small as possible to obtain a good MDL estimator. When we assume that the true distribution belongs to a certain target class which is represented by a parametric model $\mathcal{M}$, redundancy of two part codes is bounded by using another notion of regret with respect to $\mathcal{M}$. Under this assumption, a design of two part codes based on Fisher information of $\mathcal{M}$ is known [6, 7]. It is known that if $\mathcal{M}$ is an exponential family, regret of the two part code is close to minimax optimal. This fact provides a sufficiently small upper bound of redundancy. Therefore, we can obtain a good MDL estimator for exponential families by using the two part code. However, the target class $\mathcal{M}$ is often a non-exponential family in practice. Then, it is needed to develop a novel design of two part codes whose redundancy can be shown to be small for non-exponential families.

1.1.2 Contributions

In this thesis, we develop an improved method to construct two part codes such that we can show a small upper bound of their redundancy also for non-exponential families which satisfies certain conditions. The improved two part codes allow us to obtain good MDL estimators for both of exponential and non-exponential families. Then, we evaluate upper bounds of redundancy of them. By using Barron-Cover's theorem, we also evaluate upper bounds for losses of

MDL estimators induced by them. Further, we provide an example of non-exponential families to which we can apply these result directly. The example is mixture families.

Further, we discuss another example of non-exponential families, contaminated Gaussian location families. This families do not strictly satisfy some conditions we require. To overcome this problem, we introduce a kind of typical set. We can show that the conditions are satisfied under an assumption that given data belongs to the typical set. We also introduce a conditional form of Barron-Cover's theorem. By using it, we establish a performance guarantee of MDL estimators for contaminated Gaussian location families.

The third contribution is a study on Gaussian distributions. The upper bound of regret obtained for two part codes based on Fisher information we use have a term of the logarithm of the integral of the square root of the determinant of Fisher information. This term must have a finite value. To evaluate the integral, we introduce a special parametrization for Gaussian distributions whose parameter includes eigenvalues of covariance matrix directly. It is shown that a restriction of the range of eigenvalues of covariance is needed for that the integral has a finite value.

1.2 Machine learning based cyber security

Network intrusion detection systems(NIDSs) are systems which monitor network traffics and detect malicious behaviors in them. NIDSs takes an important role in cyber security. When they detect a suspicious behavior, they report an alert including information on the detected behavior. Experts of cyber security can use the alert to focus their resource on investigating important traffics.

NIDSs have their rule to detect malicious behaviors. The rules should be updated when new kinds of method of cyber attack appears to detect them. However the updating the rules usually takes a cost. It requires time and skills of experts of cyber security to design a rule for detection a new kind of attack.

Machine learning approach will be helpful. We can automatically obtain rules for detection a new kind of attack by training a machine learning model using the data including the attack.

This approach does not require experts to design new rules manually. Therefore the experts can focus their resource on practical actions on detected attacks.

In this thesis, we propose a machine learning based NIDS. Our NIDS is based on packet-level classification and consolidating them. Our experimental results show that the proposed system can achieve high classification performance.

Chapter 2

Improved MDL Estimators Using Fiber Bundles of Local Exponential Families

2.1 Introduction

The density estimation problem is investigated. Given a data string $x^n = (x_1, x_2, \cdots, x_n)$ drawn from an unknown p^*, we want to estimate p^*. We consider density estimation for parametric families, based on two-part codes in accordance with an instance of the minimum description length principle [1]. A two part code is used to compress the data string x^n, where the first part is an encoding of a density and the second part is an encoding of the data by the Shannon code using that density. In a parametric family $\mathcal{M} = \{p_\theta : \theta \in \Theta \subseteq \mathbb{R}^K\}$, a density is described by an encoding of its parameter values. We choose the density which gives the minimum code length of the two part code. In this way we can design an estimator, which is called as an MDL estimator. We examine the information theoretic regret and the statistical risk of these MDL estimators. The statistical risk of an estimator is the expectation of the loss of the estimator. We employ the Rényi divergence [4] as the loss function. It is known that if the true density is in the parametric family, then an upper bound of the risk of an MDL estimator is given by the expected regret of the two part code which induces the MDL estimator [6, 7]. Hence, if we design a two part code which has small expected regret, then the

MDL estimator enjoys a corresponding risk bound.

Through this paper, the symbol log denotes natural logarithm and we measure code length with nat. We allow code lengths to be a real value and to be considered over a non-countable set. These settings allow us to consider the Shannon code length $-\log p(x)$ for encoding x in some sample space $\mathcal{X}$ corresponding to a probability distribution p over $\mathcal{X}$. Note that in this setting, we implicitly quantize $\mathcal{X}$ independently on the distribution p and consider the code length for the quantized version. The effect of the quantization can be ignored in the comparison between distributions.

Let x^n denote a string of n samples $x^n = (x_1, x_2, \cdots, x_n) \in \mathcal{X}^n$. The regret of a code $\mathrm{REG}(p^{(n)}; \mathcal{M}, x^n)$ is defined for each x^n by the difference of its code length represented as the Shannon code length $-\log p^{(n)}(x^n)$ from the minimum of the Shannon code lengths for densities in the family $\mathcal{M}$, which corresponds to the maximum likelihood value of the parameters.

In this paper, we assume p^* belongs to the K dimensional parametric model $\mathcal{M}$. We try to obtain the risk bound of MDL estimators through evaluating their regret bounds with respect to $\mathcal{M}$. It is known that when $\mathcal{M}$ is an exponential family, by using a quantization of $\mathcal{M}$ with its precision and shape determined by the Fisher information matrix $J(\theta)$, a two part code can be constructed which enjoys the regret close to the minimax regret with respect to $\mathcal{M}$. In fact, Grünwald [7] gave an upper bound on the regret of such a code in the following form:

$$\frac{K}{2}\log n + \log \int_\Theta |J(\theta)|^{1/2} d\theta + \frac{Ka^2}{8} - K\log a + o(1), \tag{2.1}$$

where $o(1)$ is a value which converges to 0 when n goes to infinity, and a, which is an arbitrary positive constant, is a parameter of Grünwald's coding scheme. Note that the upper bound is minimized at $a = 2$.

The minimax value of the regret $\min_{p^{(n)}} \max_{x^n \in \mathcal{X}^n} \mathrm{REG}(p^{(n)}; \mathcal{M}, x^n)$, where the minimization on $p^{(n)}$ is taken over all distributions over $\mathcal{X}^n$, provides a criterion of optimality. It is known that the minimax optimal code with respect to $\mathcal{M}$ is the normalized maximum likelihood (NML) codes whose code length is $-\log p^{(n)}_{\mathrm{nml}}$, the Shannon code length corresponding to the distribution with density $p^{(n)}_{\mathrm{nml}}(x^n) \propto \max_{p \in \mathcal{M}} p(x^n)$ [8], which achieves the uniform regret for all x^n. The code length of the NML code is called the stochastic complexity of the data

with respect to $\mathcal{M}$. The NML code is the optimal in terms of the regret. However it does not induce a density estimator.

The upper bound (2.1) is close to the asymptotic evaluation of the minimax regret given in [9, 10] under certain regularity conditions, which is

$$\frac{K}{2}\log\frac{n}{2\pi}+\log\int_{\Theta}|J(\theta)|^{1/2}d\theta+o(1). \tag{2.2}$$

This means that the two part code for an exponential family is close to the optimal code in terms of regret. To obtain a risk bound of a two-part MDL estimator, we may slightly modify the code. In concrete, we simply multiple the model description length by a constant $\alpha > 1$. For $\alpha \geq 1$ (allowing for no modification with $\alpha = 1$) the regret of the modified code has an upper bound of the following form.

$$\alpha\Big(\frac{K}{2}\log n+\log\int_{\Theta}|J(\theta)|^{1/2}d\theta-K\log a\Big)+\frac{Ka^2}{8}+o(1). \tag{2.3}$$

This regret bound is also close to the minimax regret, for α near 1. This leads to a suitable upper bound for the risk of the MDL estimator.

The asymptotic regret bound (2.1) is shown only for exponential families. It would be desirable to establish two part codes for non-exponential families with the same asymptotic regret bound.

One could consider the same quantization as for exponential families. However, that would suffer an increase of the regret caused by the difference between Fisher information and empirical Fisher information, which is related to the fact that the model is not an exponential family.

To overcome this difficulty, we use a model enlargement technique using both Fisher information and empirical Fisher information, which corresponds to a fiber bundle of local exponential families [11]. We design a new two part code using the enlarged model which has the same asymptotics as the bound for the exponential family case.

Analogous concerns have arised previously [10] in the context of identifying the asymptotics of minimax regret (2.2) using Bayes codes, based on the prior of Jeffreys (with boundary modification). This Jeffreys prior works for the minimax regret for exponential families, but

to handle smooth non-exponential families as in [10, 12–14] a slightly enlarged family is used with local exponential tilting (a fiber bundle) based on the discrepancy between Fisher information and empirical Fisher information. The present paper establishes that a similar fiber bundle allows two part regret conclusion do similarly carry over from the exponential family case to the non-exponential case.

The regret bound (2.3) leads to the risk bound of the MDL estimators, owing to [2, 3, 7, 15–17], of the form of

$$\mathrm{E}_{X^n\sim p^*}\bar{d}_\lambda(p^*\|\ddot{p}_{X^n}) \le \frac{\alpha}{n}\Big(\frac{K}{2}\log n + \log\int_\Theta |J(\theta)|^{1/2}d\theta - K\log a\Big) + \frac{Ka^2}{8n} + o\Big(\frac{1}{n}\Big), \qquad (2.4)$$

where $\ddot{p}_{X^n}$ denotes the MDL estimate obtained by the data X^n and $\bar{d}_\lambda$ is the Rényi divergence of order λ (we assume $0 < \lambda < 1$) which is used as the loss function for estimators. Note that α must satisfy $\alpha \ge 1/(1-\lambda)$.

We also show that the mixture families satisfy the assumptions we need to construct the improved two part code. Then, the MDL estimator induced by the two part code for mixture families has a nice risk guarantee. A mixture family is defined as the family of convex combinations of known probability densities. These mixture families provide typical examples of non-exponential families.

The primary scope of this paper is estimation in a single parametric family, but we can use the result of this paper to solve the estimation problem for multiple families, which includes a role for the model selection. In fact, by introducing a three part code where the first part is to encode which family and then a two part code within each family, we can construct a code based on the multiple models. Then we obtain an MDL estimator based on the multiple models. The risk of such estimator is bounded by using the upper bound of the risk of the MDL estimator on each model which is provided by (2.4) with an additional term for the description of the models. Such estimator allows us to select a model as well as to estimate a density with a guarantee of the risk. This risk bound will exhibits the trade-off between complexity of the models and the accuracy of their approximation.

2.2 Preliminaries on code length and probablity distribution

In this section, we provide an introduction on the relationship with code length function and probability distributions, which is an important background of the MDL estimation.

2.2.1 Relationship between code length functions and probability distributions

Let $\mathcal{X}$ be a measurable set and $\mathcal{M}$ be a set of probability mass or density functions over $\mathcal{X}$. We call $l : \mathcal{X} \to (0, \infty)$ as a code length function over $\mathcal{X}$. This means that we consider $l(x)$ is a number of nats used to encode $x \in \mathcal{X}$. A code with the code length function l is a function which maps each $x \in \mathcal{X}$ to some nat string. We naturally require that the code is uniquely decodable. In other words, there should exists an inverse function of it. Now we assume that $\mathcal{X}$ is countable. It is a well-known fact that there exists some uniquely decodable code with the code length l if and only if the following Kraft's inequality holds:

$$\sum_{x \in \mathcal{X}} e^{-l(x)} \leq 1.$$

Let p be a (sub) probability mass function over $\mathcal{X}$. We can easily see that the code length function defined as

$$l(x) = -\log p(x)$$

satisfies the Kraft's inequality. We call the code length function defined by p as the Shannon code length corresponding to p. Conversely when we have a code length function l which satisfies the Kraft's inequality, we can always regard it as the Shannon code length corresponding to p defined by $p(x) = e^{-l(x)}$.

2.2.2 Non-integer code length

If we consider construction of a real code, code lengths of it should be an integer. However the Shannon code length $l(x) = -\log p(x)$ is not always an integer. In this thesis, we always allow such non-integer code length without rounding up.

This can be justified in terms of the average code length per sample. In concrete, we regard a probability mass function p over $\mathcal{X}$ as a probability mass function $p^{(n)}$ over $\mathcal{X}^n$ under the i.i.d. setting for each n. That is, $p^{(n)}(x^n) = \prod_{i=1}^n p(x_i)$. When we encode $x^n \in \mathcal{X}^n$, we use the Shannon code lengths $l^{(n)}(x^n) = -\log p^{(n)}(x^n) = -\sum_{i=1}^n \log p(x_i)$ with rounding up $\lceil l^{(n)}(x^n) \rceil$. This code length satisfies $l^{(n)}(x^n) \leq \lceil l^{(n)}(x^n) \rceil \leq l^{(n)}(x^n) + 1$. By dividing each side by n and taking expectation under $X^n \sim p$ i.i.d., we have $\mathrm{E}_{X\sim p} l(X) \leq \mathrm{E}_{X^n \sim p^{(n)}}(1/n)\lceil l^{(n)}(X^n) \rceil \leq \mathrm{E}_{X\sim p} l(X) + 1/n$. Therefore,

$$\lim_{n\to\infty} \frac{1}{n} \mathrm{E}_{X^n \sim p^{(n)}} \lceil l^{(n)}(X^n) \rceil = \mathrm{E}_{X\sim p} l(X)$$

holds. This implies that the difference between the expected average code length per sample based on rounding up $l^{(n)}(x^n)$ and the expected code length for single sample based on $l(x)$ without rounding can be negligible, when n is large.

2.2.3 Code length for a non-countable set

In estimation problems, the set $\mathcal{X}$ is often non-countable. Even in this case, we can work with the Shannon code length $l(x) = -\log p(x)$ defined by a probability density function p over $\mathcal{X}$. This can be justified by considering implicit quantization of $\mathcal{X}$.

Let $\{\Delta_i \mid i = 1, 2, \cdots\}$ be a cover of $\mathcal{X}$ such that $\inf_{x\in\Delta_i} p(x) > 0$ and $|\Delta_i| = \int_{\Delta_i} dx > 0$ is well defined for each i. As long as we consider only distributions with the same support, such cover can be constructed independently on the distribution p. We also take an arbitrary point $x_i \in \Delta_i$ for each i. Then we define $p_\Delta(x_i) = \inf_{x\in\Delta_i} p(x)|\Delta_i|$ for each i. This function p_Δ is a sub probability mass function over $\{x_1, x_2, \cdots\}$. Therefore there exists a uniquely decodable code with the following code length function

$$l_\Delta(x_i) = -\log p_\Delta(x_i) = -\log \inf_{x\in\Delta_i} p(x) - \log |\Delta_i|. \tag{2.5}$$

The term $-\log|\Delta_i|$ is independent on p. Therefore when we compare the code lengths between different distributions, we can ignore this term. Taking the limit as the cover $\{\Delta_i\}$ becomes infinitely fine and ignoring the second term, we have a function over $\mathcal{X}$, $l(x) = -\log p(x)$. This

function may take negative values. This is caused by the ignoring the second term in (2.5). In the context of the MDL estimation, code lengths are usually evaluated in the form of difference between two codes. Therefore for our purpose, we can regard this function l as The Shannon code length function over the non-countable set $\mathcal{X}$ even if it takes negative values.

Conversely when we have a function l satisfies

$$\int e^{-l(x)}dx \leq 1, \tag{2.6}$$

we can regard this as a Shannon code length function corresponding to a (sub) probability distribution p based on the above discussion. The inequality (2.6) is a continuous version of the Kraft's inequality.

2.3 Two part codes and MDL estimators

In this section, we review the notion of two part code and MDL estimator. Then, we review some results on the risk of MDL estimators.

An important result is the one given by [2, 3, 7, 15–17]. It states that the MDL estimator induced by a two part code has an upper bound on its statistical risk which equals the redundancy of the code, that is, when the redundancy of the two part code is small, then the upper bound on the risk of the MDL estimator is small. Hence, to obtain a good MDL estimator, we should construct a two part code which enjoys a small redundancy.

We note that the original form of risk bound in [2] is given by the index of resolvability, which is an upper bound on redundancy and is related to the trade-off between complexity of the models and the accuracy of their approximation.

Let $\mathcal{X}$ be a measurable set, $\mathcal{M} = \{p_\theta : \theta \in \Theta\}$ a parametric model of probability densities over $\mathcal{X}$, and $\{\ddot{\mathcal{P}}_n\}$ a sequence of countable sets of probability densities over $\mathcal{X}$. Usually each $\ddot{\mathcal{P}}_n$ is assumed to be a subset of $\mathcal{M}$, but it is not always assumed. For a probability density p over $\mathcal{X}$, we let $p(x^n) = \prod_{t=1}^n p(x_t)$, i.e. we assume x^n is i.i.d. Further, for each $n \in \{1, 2, \ldots\}$, let L_n be a code length function satisfying the Kraft's inequality on $\ddot{\mathcal{P}}_n$. For a fixed $\alpha \geq 1$, we define

a code length function $L_{\rm tp}^{(n)}$ on $\mathcal{X}^n$ as

$$L_{\rm tp}^{(n)}(x^n) = \min_{p\in\ddot{\mathcal{P}}_n}\Big(-\log p(x^n) + \alpha L_n(p)\Big). \tag{2.7}$$

Since $e^{-L_{\rm tp}^{(n)}(x^n)} \le \sum_{p\in\ddot{\mathcal{P}}_n} p(x^n)e^{-\alpha L_n(p)}$, we can see that $L_{\rm tp}^{(n)}$ satisfies Kraft's inequality on $\mathcal{X}^n$ (2.6).

When $x^n \in \mathcal{X}^n$ is given, let $\ddot{p} = \ddot{p}_{x^n}$ denote the minimizer of (2.7), i.e.

$$\ddot{p} = \ddot{p}_{x^n} = \arg\min_{p\in\ddot{\mathcal{P}}_n}\Big(-\log p(x^n) + \alpha L_n(p)\Big).$$

Note that $\ddot{p}(x^n)$ always means $\ddot{p}_{x^n}(x^n)$. Then, we can interpret $L_{\rm tp}^{(n)}$ as the codelength of a two part code, i.e. we encode $\ddot{p}$ with the code length $\alpha L_n(\ddot{p})$, then we encode x^n by the Shannon code corresponding to $\ddot{p}$. Let $p_{\rm tp}^{(n)} = p_{\rm twopart}^{(n)}$ denote the sub probability distribution over $\mathcal{X}^n$ corresponding to $L_{\rm tp}^{(n)}$ as

$$p_{\rm tp}^{(n)}(x^n) = e^{-L_{\rm tp}^{(n)}(x^n)} = \ddot{p}(x^n)e^{-\alpha L_n(\ddot{p})}.$$

Remark: We have used the superscript (n) in notations like $L_{\rm tp}^{(n)}$ and $p_{\rm tp}^{(n)}$ to emphasize the fact that they are functions over $\mathcal{X}^n$ not over $\mathcal{X}$. In particular, we remark that $p_{\rm tp}^{(n)}(x^n)$ does not mean the i.i.d. density represented as a production of densities given by a distribution over $\mathcal{X}$. We often use the same superscript when we consider notions related to $\mathcal{X}^n$ or distributions over $\mathcal{X}^n$.

The code length function $L_{\rm tp}^{(n)}$ depends on $(\ddot{\mathcal{P}}_n, \alpha L_n)$ for each n. Hence in this paper, we call a code with code length function $L_{\rm tp}^{(n)}$ as "two part code $L_{\rm tp}^{(n)}$ based on $(\ddot{\mathcal{P}}_n, \alpha L_n)$". We call the function which maps x^n to $\ddot{p}_{x^n}$ as the MDL estimator defined by the two part code $L_{\rm tp}^{(n)}$ based on $(\ddot{\mathcal{P}}_n, \alpha L_n)$.

It is known that a two part code with small redundancy gives a good MDL estimator. To state it, here we give the definition of Rényi divergence and redundancy. Rényi divergence of degree $\lambda \in (0,1)$, which is a measure of differences between two probability densities, between p and q defined as

$$\bar{d}_\lambda(p\|q) = -\frac{1}{1-\lambda}\log \mathrm{E}_p\Big(\frac{q(X)}{p(X)}\Big)^{1-\lambda}.$$

The Rényi divergence is an increasing and continuous function of λ, and the limit $\lim_{\lambda\to 1}\bar{d}_\lambda(p\|q)$ equals the Kullback-Leibler divergence below.

$$D(p\|q) = \mathrm{E}_p \log \frac{p(X)}{q(X)}.$$

Define the redundancy of a Shannon code corresponding to $p^{(n)}$ with respect to p^* as

$$\mathrm{RED}^{(n)}(p^*, p^{(n)}) = \mathrm{E}_{X^n\sim p^*}\Big[\log \frac{p^*(X^n)}{p^{(n)}(X^n)}\Big].$$

This is same as the Kullback-Leibler divergence $D(p^*\|p^{(n)})$ when we regard p^* as a distribution over X^n under the i.i.d. setting. We define

$$\overline{\mathrm{RED}}(n) = \sup_{p^*\in\mathcal{M}} \mathrm{RED}^{(n)}(p^*, p^{(n)}_{\mathrm{tp}}),$$

if it is finite. When

$$\lim_{n\to\infty} \frac{\overline{\mathrm{RED}}(n)}{n} = 0, \tag{2.8}$$

the two part code is a universal code for $\mathcal{M}$.

The following theorem holds.

Theorem 2.1 (Barron and Cover 1991, Li 1999, Zhang 2004, Grünwald 2007)**.** *Let p^* be an arbitrary probability density over $\mathcal{X}$. For arbitrary $\alpha > 1, \lambda \in (0, 1-\alpha^{-1}], \ddot{\mathcal{P}}_n$ and L_n, the MDL estimator defined by $(\ddot{\mathcal{P}}_n, \alpha L_n)$ satisfies the following inequalities.*

$$\mathrm{E}_{X^n\sim p^*}\bar{d}_\lambda(p^*\|\ddot{p}_{X^n}) \le \frac{1}{n}\mathrm{RED}^{(n)}(p^*, p^{(n)}_{\mathrm{tp}}) \le \min_{\bar{p}\in\ddot{\mathcal{P}}_n}\Big(D(p^*\|\bar{p}) + \frac{1}{n}L_n(\bar{p})\Big). \tag{2.9}$$

The last side of (2.9) is the index of resolvability. We provide a proof in Section 2.9.1.

If (2.8) holds, Theorem 2.1 implies that the risk of the MDL estimator defined by this two part code converges to 0, uniformly for $p^* \in \mathcal{M}$. This kind of result on MDL estimators first appeared in [2], where the bound is given as the index of resolvability and the squared Hellinger distance $d^2_H(p^*, p) = \int(\sqrt{p^*(x^n)} - \sqrt{p(x)})^2 dx$ is used instead of the Rényi divergence. That theorem and its proof have been improved in [15] and [16]. Theorem 2.1 is of the form

in [7]. The book [7] also provides a historical note on this theorem. The bound in Theorem 2.1 includes the bound in terms of the squared Hellinger distance, since $d_H^2(p^*, p) \leq \bar{d}_{1/2}(p^* \| p)$ holds. The bound in terms of α-divergence $D_\alpha(p^* \| p) = 4/(1-\alpha^2)\mathrm{E}_{X\sim p^*}(1-(p(x)/p^*(x))^{(1+\alpha)/2})$ is also included, since $D_{1-2\lambda}(p^* \| p) \leq \lambda^{-1}\bar{d}_\lambda(p^* \| p)$ holds.

A probabilistic performance guarantee is also known [3, 18].

Theorem 2.2 (Zhang 2006, Chatterjee and Barron 2014)**.** *Let p^* be an arbitrary distribution on $\mathcal{X}$. For arbitrary $\alpha > 1, \lambda \in (0, 1-\alpha^{-1}], b > 0, \ddot{\mathcal{P}}_n$ and L_n, the MDL estimator defined by $(\ddot{\mathcal{P}}_n, \alpha L_n)$ satisfies the following inequality.*

$$\Pr\Big\{\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) > \frac{1}{n}\log\frac{p^*(X^n)}{p_{\mathrm{tp}}^{(n)}(X^n)} + b\Big\} < e^{-n\alpha^{-1}b},$$

where Pr *is measured with $X^n \sim p^*$.*

This theorem provides an upper bound, which is exponentially small in n, on the probability of the event that the loss exceeds the value

$$-\log p_{\mathrm{tp}}^{(n)}(x^n) - (-\log p^*(x^n))$$

by b.

We provide a proof in Section 2.9.2.

Theorems 2.1 and 2.2 imply that a two part code which enjoys a small regret provides an MDL estimator with small loss (with a high probability) and risk, because a code with small regret shows a small redundancy as well. Hence we pursue a two-stage code with small regret for non-exponential families.

The regret of a code p with respect to $\mathcal{M}$ and x^n is defined by

$$\mathrm{REG}(p; \mathcal{M}, x^n) = \max_{q\in\mathcal{M}}\Big\{-\log p(x^n) - \Big(-\log q(x^n)\Big)\Big\}.$$

Then, since the maximum over $q \in \mathcal{M}$ is greater than at $p^* \in \mathcal{M}$,

$$\mathrm{RED}^{(n)}(p^*, p) \leq \mathrm{E}_{p^*}\mathrm{REG}(p; \mathcal{M}, x^n)$$

holds.

In this paper, we assume that $\mathcal{M}$ is a signle parametric model $\mathcal{M} = \{p_\theta : \theta \in \Theta \subseteq \mathbb{R}^K\}$ ($K \geq 1$). Grünwald [7] provides a way of constructing a two part code which achieves the regret bound (2.1). It is close to the minimax value (2.2), for exponential families. In the next section, we will introduce the way of this construction and show its properties which lead to a nice regret bound. Further, in the section 2.5, we will develop an extension of this method so that we can obtain a small upper bound on the regret for non exponential families.

2.4 Grünwald's construction of two part codes for exponential families

In this section, we construct a two part code $(\ddot{\Theta}_n, L_n)$ which is appropriate for our goal when $\mathcal{M}$ is an exponential family, following Grünwald [7]. Then, we give an analysis on the code length of the two part code, which provides the upper bound of the regret of the form (2.3) close to the minimax regret. This analysis is finer than that of [7] and gives a bound for finite sample size. In concrete, [7] provides only the asymptotic regret bound with $o(1)$ notation (2.3) and the strict form of the $o(1)$ term is not shown. Our analysis provides the regret bound of the form without $o(1)$ notation,

$$\frac{1}{\alpha}\mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \leq \frac{K}{2}\log n + \log \int_\Theta |J(\theta)|^{1/2} d\theta + \frac{C_n K a^2}{8\alpha} - K \log a + r(n). \quad (2.10)$$

The C_n and $r(n)$ are determined by some constants related to $\mathcal{M}$ and $\lim_{n\to\infty} C_n = 1$ and $\lim_{n\to\infty} r(n) = 0$ hold. This bound asymptotically corresponds to the bound with $o(1)$ notation (2.3). However we provide the concrete forms of C_n and $r(n)$. Therefore our bound (2.10) holds for finite sample size n. This regret bound is shown later in Lemma 4 formally.

In this paper, we assume $\mathcal{M}$ is a K dimensional parametric model

$$\mathcal{M} = \{p_\theta : \theta \in \Theta\},$$

where Θ is a convex and compact subset of $\mathbb{R}^K$. We denote the maximum likelihood estimate with respect to Θ as

$$\hat{\theta} = \hat{\theta}(x^n) = \arg\max_{\theta\in\Theta} p_\theta(x^n). \quad (2.11)$$

In this setting, we may assume that Θ is a subset of a naturally determined parameter space $\tilde{\Theta}$. Here, "naturally determined" means that the space cannot be extended any more. For example, the probability parameter for the Bernoulli model has the natural range $[0, 1]$ and Θ may be an interval contained in $[0, 1]$. Then, when $\hat{\theta}$ is on the boundary of Θ, $\hat{\theta}$ is not always a stationary point of the likelihood function over $\tilde{\Theta}$. In that case, the essential maximum likelihood estimate, which is with respect to $\tilde{\Theta}$, is in $\tilde{\Theta} \setminus \Theta$. However, in this paper, we always let the maximum likelihood estimate refer to (2.11) which is with respect to the restriction Θ.

First, we put some assumptions on the model $\mathcal{M}$ for our analysis. We assume the differentiability of the likelihood function.

Assumption 1. *For all x and for all $\theta \in \Theta$, $\log p_\theta(x)$ is 3 times continuously differentiable with respect to θ.*

Define the empirical Fisher information matrix by

$$\hat{J}_{ij}(\theta; x^n) = -\frac{1}{n} \sum_{t=1}^{n} \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log p_\theta(x_t)$$

and let $\hat{J}(\theta; x)$ denote it for the case of $n = 1$. Further, we define the Fisher information matrix $J(\theta)$ as the expectation of $\hat{J}(\theta; x)$ as $J(\theta) = \mathrm{E}_{p_\theta} \hat{J}(\theta; X)$. Then, we assume that for all $\theta \in \Theta$, $J(\theta)$ is positive definite and the eigenvalues of $J(\theta)$ are bounded on Θ.

Assumption 2. *There exist positive ζ and $\bar{\lambda}$ such that for all unit vectors z and for all $\theta \in \Theta$, the following holds.*

$$\zeta \le z^T J(\theta) z \le \bar{\lambda}.$$

Assumption 3. *The determinant $|J(\theta)|$ is a differentiable function on Θ.*

Assumption 4. *There exists $D_J > 0$ such that*

$$\sup_{|z|=1, \theta \in \Theta} \left| \nabla_z |J(\theta)|^{1/2} \right| \le D_J,$$

where ∇_z means the directional derivative with respect to θ to the direction of z.

Assumption 5. *There exist* $\kappa > 0$ *and* $\bar{b} > 0$ *such that for all* $(\theta_1, \theta_2) \in \Theta^2$ *with* $|\theta_1 - \theta_2| \leq \bar{b}$*, the following holds.*

$$\max_{z \in \mathbb{R}^K \setminus \{0\}} \frac{z^T J(\theta_1) z}{z^T J(\theta_2) z} \leq 1 + \kappa |\theta_1 - \theta_2|.$$

To handle the sequences x^n for which the maximum likelihood estimate is on the boundary of Θ, denoted by $\partial\Theta$, we use the following assumption.

Assumption 6. *It is possible to construct a two part code* $L_{\mathrm{tp}}^{(n)\partial}(x^n) = -\log p_{\mathrm{tp}}^{(n)\partial}(x^n)$ *over* $\{x^n : \hat{\theta}(x^n) \in \partial\Theta\}$ *based on* $(\ddot{\mathcal{P}}_n^{\partial}, \alpha L_n^{\partial})$*, such that, for a certain* $d > 0$*, for all large n, for all* $\alpha > 0$*, for all* x^n *with* $\hat{\theta}(x^n) \in \partial\Theta$*,*

$$L_{\mathrm{tp}}^{(n)\partial}(x^n) - \log \frac{1}{p_{\hat{\theta}}(x^n)} \leq \frac{\alpha(K-d)}{2} \log n + o(\log n)$$

holds.

Remark: If $\partial\Theta$ consists of hyper surfaces in $\mathbb{R}^K$, for example a surface of a polyhedron or that of a sphere, it is easy to construct such $p_{\mathrm{tp}}^{(n)\partial}$ with $d = 1$.

The code we investigate uses a quantization of Θ based on the Fisher information matrix. This quantization, which we quote from Grünwald (Chapter 10 of [7]), originates with the analysis by Barron [6].

Let $\ddot{\Theta}_n \subseteq \Theta$ be a quantized version of Θ and $\ddot{\mathcal{P}}_n = \{p_\theta : \theta \in \ddot{\Theta}_n\}$. Further, let L_n be a code length function on $\ddot{\Theta}_n$. When $\mathcal{M}$ is an exponential family of densities, we can show a small upper bound on the regret of the two part code.

Fix $a > 0$. Here we introduce the construction of the two part code in [7]. First, we partition each axis of $\mathbb{R}^K$ into $\{[(i-1)an^{-\beta}, ian^{-\beta})\}_{i \in \mathbb{Z}}$ with $0 < \beta < 1/2$. This partition defines hyper cubes with side lengths $an^{-\beta}$. Note that Grünwald assumes $\beta = 1/4$, which is turned to provide the best bound later in our analysis.

We construct a cover of Θ which consists of these hyper cubes. We call the intersections of Θ and these hyper cubes large cells. For each x^n, we define $\theta_S = \theta_S(x^n)$ as the center of mass of the large cell which includes $\hat{\theta}(x^n)$. Then, for all x^n, if θ belongs to the large cell including

$\theta_S(x^n)$, then

$$|\theta - \theta_S(x^n)| \leq \frac{\sqrt{K}a}{2} n^{-\beta}$$

holds. For each large cell $\mathfrak{S}$, let $\theta_{\mathfrak{S}}$ be the center of mass of $\mathfrak{S}$. Since the Fisher information matrix $J(\theta_{\mathfrak{S}})$ is positive definite by Assumption 2, we can define the ellipsoid

$$\left\{\theta : (\theta - \theta_{\mathfrak{S}})^T J(\theta_{\mathfrak{S}})(\theta - \theta_{\mathfrak{S}}) \leq \frac{a^2}{4n}\right\} \tag{2.12}$$

with the center $\theta_{\mathfrak{S}}$.

For each large cell $\mathfrak{S}$, consider the hyper rectangle which circumscribes the ellipsoid (2.12) and whose edges are parallel to it there axes. This is a small hyper rectangle centered in $\mathfrak{S}$. Next we construct a cover of each $\mathfrak{S}$ using disjoint copies of that hyper rectangle with center shifted to cover $\mathfrak{S}$.

We call the intersection of Θ and each of the hyper rectangles a small cell. Then for each small cell, we put a quantized point on its center of mass. Let $\ddot{\Theta}_n$ denote the set of all such quantized points. Then, define L_n as the code length of the fixed length code for $\ddot{\Theta}_n$:

$$L_n(\theta) = \log|\ddot{\Theta}_n|.$$

Now, define the MDL estimator induced by our $(\ddot{\Theta}_n, \alpha L_n)$ as

$$\ddot{\theta} = \ddot{\theta}(x^n) = \arg\min_{\theta \in \ddot{\Theta}_n} \Big(-\log p_\theta(x^n) + \alpha L_n(\theta) \Big),$$

which is the point corresponding to the MDL estimate $\ddot{p}$. Then, the code length of the two part code is

$$-\log p^{(n)}_{\mathrm{tp}}(x^n) = -\log p_{\ddot{\theta}}(x^n) + \alpha L_n(\ddot{\theta}).$$

Under Assumptions 1 to 5, we can show the following proposition and lemma

Proposition 1. *Suppose Assumptions 1 to 5 hold. Let $\ddot{\hat{\theta}}$ denote any of the closest points in $\ddot{\Theta}_n$ to $\hat{\theta}$. The quantization $\ddot{\Theta}_n$ satisfies the following inequalities for all x^n.*

$$(\ddot{\hat{\theta}} - \hat{\theta})^T J(\theta_S)(\ddot{\hat{\theta}} - \hat{\theta}) \leq \frac{Ka^2}{4n}. \tag{2.13}$$

Further, for an arbitrary point on the line θ between $\ddot{\hat{\theta}}$ and $\hat{\theta}$, we have

$$|\theta - \theta_S| \leq \sqrt{K} a n^{-1/4}. \tag{2.14}$$

Lemma 1. *Suppose Assumptions 1 to 5 hold. The code length function $L_n(\theta) = \log|\ddot{\Theta}_n|$ on $\ddot{\Theta}_n$ satisfies the following inequality for all $n \geq (\sqrt{K}a/\bar{b})^{1/\beta}$ and for all $\theta \in \ddot{\Theta}_n$,*

$$L_n(\theta) \leq \frac{K}{2}\log n + \log \int_\Theta |J(\theta')|^{1/2} d\theta' - K \log a + r_\beta(n),$$

where

$$r_\beta(n) = \log(1+C_J n^{-(1/2-\beta)}) + \log(1+C_\Theta n^{-\beta}) + \log(1+C_{J,K} n^{-\beta}), \tag{2.15}$$

and C_J, C_Θ and $C_{J,K}$ are the constants provided in the proof.

The proofs of the proposition and the lemma stated above are given in Section 2.6. Note that $r_\beta(n)$ is decreasing and $\lim_{n\to\infty} r_\beta(n) = 0$. We see in this expression for $r_\beta(n)$ that $\beta = 1/4$ provides the best bound with $r(n) = O(n^{-1/4})$.

The following lemmas describe properties of the quantized space $\ddot{\Theta}_n$.

Lemma 2. *Let $\ddot{\hat{\theta}}$ denote any of the closest points in $\ddot{\Theta}_n$ to $\hat{\theta}$. Under Assumption 5, for all $n \geq (\sqrt{K}a/\bar{b})^{1/\beta}$,*

$$\sup_{x^n:\hat{\theta}\in\Theta^\circ} (\ddot{\hat{\theta}} - \hat{\theta})^T J(\hat{\theta})(\ddot{\hat{\theta}} - \hat{\theta}) \leq \frac{Ka^2}{4n}(1 + \kappa\sqrt{K} a n^{-\beta}). \tag{2.16}$$

Proof of Lemma 2. Since $\hat{\theta}$ and θ_S belong to the same large cell with side length $an^{-\beta}$,

$$|\hat{\theta} - \theta_S| \leq \sqrt{K} a n^{-\beta}$$

holds. Therefore, from Assumption 5, when $n \geq (\sqrt{K}a/\bar{b})^{1/\beta}$, we have for all θ which belongs to the same large cell as $\hat{\theta}$,

$$(\theta - \hat{\theta})^T J(\hat{\theta})(\theta - \hat{\theta}) \leq (\theta - \hat{\theta})^T J(\theta_S)(\theta - \hat{\theta})\Big(1 + \kappa\sqrt{K} a n^{-\beta}\Big). \tag{2.17}$$

From the fact that $\hat{\theta}$ and $\ddot{\hat{\theta}}$ belong to the same small cell defined by using $J(\theta_S)$,

$$(\ddot{\hat{\theta}} - \hat{\theta})^T J(\theta_S)(\ddot{\hat{\theta}} - \hat{\theta}) \leq \frac{Ka^2}{4n}$$

holds. By plugging in $\ddot{\hat{\theta}}$ for θ in (2.17), we have (2.16). □

Lemma 3. *Under Assumptions 4 and 5, the following inequality holds for all* $n \geq (\sqrt{K}a/\bar{b})^{1/\beta}$, *for all* x^n, *and for all* θ' *between* $\ddot{\hat{\theta}}$ *and* $\hat{\theta}$.

$$(\ddot{\hat{\theta}} - \hat{\theta})^T J(\theta')(\ddot{\hat{\theta}} - \hat{\theta}) \leq \frac{C_n Ka^2}{4n},$$

where

$$C_n = (1 + \kappa\sqrt{K}an^{-\beta})(1 + \kappa\sqrt{K}an^{-1/2}\zeta^{-1/2}/2). \tag{2.18}$$

Proof of Lemma 3. By using Assumption 5 and Lemma 2, we have

$$(\ddot{\hat{\theta}} - \hat{\theta})^T J(\theta')(\ddot{\hat{\theta}} - \hat{\theta}) \leq \frac{Ka^2}{4n}(1 + \kappa\sqrt{K}an^{-\beta})(1 + \kappa|\theta' - \hat{\theta}|).$$

Since θ' and $\hat{\theta}$ belong to the same small cell, from Assumption 2, $|\theta' - \hat{\theta}|^2 \leq Ka^2/4n\zeta$ holds. Then,

$$(\ddot{\hat{\theta}} - \hat{\theta})^T J(\theta')(\ddot{\hat{\theta}} - \hat{\theta}) \leq \frac{Ka^2}{4n}(1 + \kappa\sqrt{K}an^{-\beta})(1 + \kappa\sqrt{K}an^{-1/2}\zeta^{-1/2}/2).$$

□

Since $\ddot{\hat{\theta}}$ may not be the minimum solution to the description length, the regret of the two part code is bounded by

$$\mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \leq \log \frac{p_{\hat{\theta}}(x^n)}{p_{\ddot{\hat{\theta}}}(x^n)} + \alpha L_n(\ddot{\hat{\theta}}).$$

By the Taylor's theorem, for x^n with $\hat{\theta} \in \Theta^\circ$, there exists θ' between $\ddot{\hat{\theta}}$ and $\hat{\theta}$ such that

$$\log \frac{p_{\hat{\theta}}(x^n)}{p_{\ddot{\hat{\theta}}}(x^n)} = \frac{n}{2}(\ddot{\hat{\theta}} - \hat{\theta})^T \hat{J}(\theta'; x^n)(\ddot{\hat{\theta}} - \hat{\theta}). \tag{2.19}$$

Then, we have

$$\mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \le \frac{n}{2}(\ddot{\hat{\theta}} - \hat{\theta})^T \hat{J}(\theta'; x^n)(\ddot{\hat{\theta}} - \hat{\theta}) + \alpha L_n(\ddot{\hat{\theta}}). \tag{2.20}$$

When $\mathcal{M}$ is an exponential family and θ is its canonical parameters, $\hat{J}(\theta; x^n) = J(\theta)$ holds for all $\theta \in \Theta$ and for all x^n. Using this fact, we can show the following lemma from Lemma 1 and Lemma 3.

Lemma 4. *When $\mathcal{M}$ is an exponential family and Θ is the range of its canonical parameters, our two part code has the regret which satisfies the following. For all x^n with $\hat{\theta}(x^n) \in \Theta^\circ$,*

$$\frac{1}{\alpha}\mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \le \frac{K}{2}\log n + \log \int_\Theta |J(\theta)|^{1/2} d\theta + f(n), \tag{2.21}$$

where

$$f(n) = \frac{C_n K a^2}{8\alpha} - K \log a + r(n),$$

$r(n)(= O(n^{-1/4}))$ denotes $r_\beta(n)$ (2.15) with $\beta = 1/4$, and $C_n(= 1 + o(1))$ is defined as (2.18).

The proofs of Proposition 1 and Lemma 1 and Lemmas 2 and 3 will be provided in Section 2.6.

2.4.1 Note on boundary problem

The code $p_{\mathrm{tp}}^{(n)}$ is good for x^n with $\hat{\theta} \in \Theta^\circ$, but (2.21) is not guaranteed for x^n with $\hat{\theta} \notin \Theta^\circ$. That is, $p_{\mathrm{tp}}^{(n)}$ is not enough for our purpose, and we need a bound for all x^n. For this point, important is Assumption 6, which requires existence of a code $p_{\mathrm{tp}}^{(n)\partial}$ with small regret for x^n with $\hat{\theta} \in \partial\Theta$. That is, under Assumption 6, we can modify $p_{\mathrm{tp}}^{(n)}$ with the help from $p_{\mathrm{tp}}^{(n)\partial}$, so that the regret bound (2.21) holds also when $\hat{\theta} \in \partial\Theta$.

Let $l_1(n) = n^{-\iota}$ with $d > 2\iota > 0$. Define

$$-\log p'^{(n)}_{\mathrm{tp}}(x^n) = \begin{cases} -\log p_{\mathrm{tp}}^{(n)}(x^n) + \alpha l_1(n), & \text{if } \hat{\theta} \in \Theta^\circ, \\ -\log p_{\mathrm{tp}}^{(n)\partial}(x^n) - \alpha \log(1 - e^{-l_1(n)}), & \text{if } \hat{\theta} \in \partial\Theta. \end{cases} \tag{2.22}$$

In this scheme, $p'^{(n)}_{\mathrm{tp}}(x^n)$ employs $p^{(n)}_{\mathrm{tp}}$ when $\hat{\theta} \in \Theta^\circ$, otherwise it uses $p^{(n)\partial}_{\mathrm{tp}}$. The terms $\alpha l_1(n)$ and $-\alpha \log(1 - e^{-l_1(n)})$ are extra code length to indicate whether we use $p^{(n)}_{\mathrm{tp}}$ or $p^{(n)\partial}_{\mathrm{tp}}$. We employ the MDL estimator defined by $p'^{(n)}_{\mathrm{tp}}$ for our purpose. Note that since $l_1(n)$ depends only on n, MDL estimator outputs the same estimate as that by $p^{(n)}_{\mathrm{tp}}$, when $\hat{\theta} \in \Theta^\circ$, and the same one as by $p^{(n)\partial}_{\mathrm{tp}}$, otherwise.

Now assume that n satisfies $l_1(n) = n^{-\iota} \leq 1/2$ (i.e. $n \geq 2^{1/\iota}$), then $e^{-l_1(n)} \leq 1 - l_1(n)/2$ holds. Then, when $n \geq 2^{1/\iota}$, the following holds.

$$-\log(1 - e^{-l_1(n)}) \leq -\log(1 - 1 + l_1(n)/2) = \iota \log n + \log 2,$$

which implies that, if $\hat{\theta} \in \partial\Theta$,

$$\log \frac{p_{\hat{\theta}}(x^n)}{p'^{(n)}_{\mathrm{tp}}(x^n)} = \log \frac{p_{\hat{\theta}}(x^n)}{p^{(n)\partial}_{\mathrm{tp}}(x^n)} + \alpha \log(1 - e^{-l_1(n)}) \leq \alpha\Big(\frac{K - (d - 2\iota)}{2} \log n + \log 2\Big),$$

holds. As for the case that $\hat{\theta} \in \Theta^\circ$, we have

$$\log \frac{p_{\hat{\theta}}(x^n)}{p'^{(n)}_{\mathrm{tp}}(x^n)} \leq \alpha\Big(\frac{K}{2} \log n + \log \int_\Theta |J(\theta)|^{1/2} + f(n) + l_1(n)\Big).$$

Here, the former is smaller than the latter, when n is large. (Recall $2\iota < d$ is assumed.) Let

$$\overline{\mathrm{REG}}(n) = \alpha\Big(\frac{K}{2} \log n + \log \int_\Theta |J(\theta)|^{1/2} d\theta + f(n) + l_1(n)\Big). \tag{2.23}$$

Then for large n, $\mathrm{REG}(p'^{(n)}_{\mathrm{tp}}, \mathcal{M}, x^n) \leq \overline{\mathrm{REG}}(n)$ holds for all x^n. Therefore, by Theorems 2.1 and 2.2, we have the following.

Theorem 2.3 (Bounds for exponential families)**.** *Let $\mathcal{M}$ be an exponential family which satisfies Assumptions 1–6. Let $\ddot{p}_{X^n}$ be the estimate by the MDL estimator induced by $p'^{(n)}_{\mathrm{tp}}$. When X^n is drawn from $p^* \in \mathcal{M}$, for large n, we have*

$$\mathrm{E}_{X^n \sim p^*} \bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) \leq \frac{1}{n} \overline{\mathrm{REG}}(n)$$

and

$$\Pr\Big\{\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) > \frac{1}{n} \overline{\mathrm{REG}}(n) + b\Big\} < e^{-n\alpha^{-1}b},$$

where b is an arbitrary positive real and Pr *is defined for $X^n \sim p^*$.*

Remark: The analysis may be generalized to two part codes with possible non-compact parameter spaces taking advantage of a continuous prior probability density function $w(\theta)$. The parameter space may be split into moderately small regions such that within each region, by continuity, $w(\theta)$ and $|J(\theta)|$ are nearly constant (these regions being described with length the minus log of their prior probability). These regions are further quantized into the small cells discussed above (described with length the log cardinality of them in each region). Then the regret $L^{(n)}_{\mathrm{tp}}(x^n) - \log(1/p_{\hat{\theta}}(x^n))$ in an exponential family has the bound

$$\frac{K}{2}\log n + \log\frac{|\hat{J}(\hat{\theta})|^{1/2}}{w(\hat{\theta})} + \frac{Ka^2}{8} - K\log a + O\Big(\frac{1}{n}\Big).$$

In the case that $\int |J(\theta)|^{1/2}d\theta$ is finite the conclusion of Lemma 4 arises with the choice of Jeffreys prior

$$w(\theta) = \frac{|J(\theta)|^{1/2}}{\int |J(\theta)|^{1/2}d\theta}.$$

2.5 Extension to Non-exponential families

We have given the upper bound (2.23) of the regret for exponential families which is close to optimal. What about non-exponential families? We find that for non-exponential families, one cannot obtain similar upper bound by the same two part code as for exponential-families. However, we can improve this two part code using a fiber bundle of local exponential families [11] so that we can obtain a regret bound similar to (2.1) under some assumptions.

The inequality (2.20) itself still holds for non-exponential families, but $\hat{J}(\theta; x^n)$ differs from $J(\theta)$ in general for any parameterization. In particular it does even at $\theta = \hat{\theta}$. Therefore, we have to deal with the difference between them to evaluate an upper bound on $(\ddot{\theta} - \hat{\theta})^T \hat{J}(\theta; x^n)(\ddot{\theta} - \hat{\theta})$ for non-exponential families. In this section we describe our recipe based on fiber bundle of local exponential families for the problem.

2.5.1 Fiber Bundle of Local Exponential Families

The fiber bundle of local exponential families was introduced by Amari and Nagaoka [11]. It provides a general method to obtain an enlarged model in the form of a fiber bundle whose

fibers are the exponential family with respect to additional parameters. We use the enlarged model as a model on which our two part code based. We show that this approach helps us to solve the problem caused by the difference between $\hat{J}(\theta; x^n)$ and $J(\theta)$.

Following [12], we define a random variable $V(\theta; x^n)$ which represents the difference between Fisher information and empirical Fisher information:

$$V(\theta; x^n) = J^{-1/2}(\theta)^T \hat{J}(\theta; x^n) J^{-1/2}(\theta) - I,$$

where I is the unit matrix of order K. Note that $\mathrm{E}_{p_\theta} V(\theta; x^n) = J^{-1/2} J J^{-1/2} - I = 0$. Let $\Xi_0 = [-\xi_0, \xi_0]^{K \times K}$ for a constant $\xi_0 > 0$. For each $\theta \in \Theta$ and each $\xi \in \Xi_0$, we define the following density:

$$\bar{p}_{\theta,\xi}(x) = p_\theta(x) \exp\Big(\xi \cdot V(\theta; x) - \psi_\theta(\xi)\Big),$$
$$\psi_\theta(\xi) = \log \int p_\theta(x) \exp\Big(\xi \cdot V(\theta; x)\Big) dx.$$

Here, $\xi \cdot V(\theta; x)$ denotes the Frobenius inner product of ξ and $V(\theta; x)$. Let $\bar{p}_{\theta,\xi}(x^n) = \prod_{t=1}^n \bar{p}_{\theta,\xi}(x_t)$. Note that

$$\bar{p}_{\theta,\xi}(x^n) = p_\theta(x^n) \exp\Big(n(\xi \cdot V(\theta; x^n) - \psi_\theta(\xi))\Big) \tag{2.24}$$

holds. For each θ, the family $\mathcal{M}_e(\theta) = \{\bar{p}_{\theta,\xi} : \xi \in \Xi_0\}$ is an exponential family with the natural parameter ξ. When $\xi = 0$, $\bar{p}_{\theta,\xi}$ equals p_θ. The structure where for each θ, the exponential family $\mathcal{M}_e(\theta)$ is supplemented to $\mathcal{M}$ forms a fiber bundle $\bar{\mathcal{M}} = \{\bar{p}_{\theta,\xi} \mid (\theta, \xi) \in \Theta \times \Xi_0\}$. Fiber bundles of this form is called as a fiber bundle of local exponential families. We use $\bar{\mathcal{M}}$ to construct our two part code.

2.5.2 Two-Part Codes based on Fiber Bundle of Local Exponential Families

We consider a finite subset $\Xi_n \subset \Xi_0$, a quantized version of Ξ_0. Let $\tilde{L}_n(\xi)$ be a code length function on Ξ_n satisfying Kraft's inequality. Let $\ddot{\Theta}_n$ and L_n be the same one as what we constructed in the previous section. We employ a parameter space $\ddot{\Theta}_n \times \Xi_n$ for the two part code using $\{\bar{p}_{\theta,\xi}\}$. Let $\bar{L}_n(\theta, \xi) = L_n(\theta) + \tilde{L}_n(\xi)$. The two part code is denoted as $(\ddot{\Theta}_n \times \Xi_n, \alpha \bar{L}_n)$.

Given the two part code, we define an MDL estimator as

$$(\ddot{\theta}, \ddot{\xi}) = \arg \min_{(\theta,\xi)\in\ddot{\Theta}_n\times\Xi_n} \Big(-\log \bar{p}_{\theta,\xi}(x^n) + \alpha \bar{L}_n(\theta, \xi)\Big).$$

Note that the estimate $\bar{p}_{\ddot{\theta},\ddot{\xi}}$ may be a distribution outside $\mathcal{M}$. Define a sub probability distribution corresponding to the modified two part code as

$$\bar{p}^{(n)}_{\mathrm{tp}}(x^n) = \bar{p}_{\ddot{\theta},\ddot{\xi}}(x^n) e^{-\alpha \bar{L}_n(\ddot{\theta},\ddot{\xi})}.$$

We construct Ξ_n so that the new two part code has a good upper bound of its regret. Define the matrix $\mathcal{E}^{(l,m)}$ for each (l, m) as

$$[\mathcal{E}^{(l,m)}]_{ij} = \mathbb{1}((i, j) = (l, m)),$$

where $\mathbb{1}(\cdot)$ is the indicator function. Then, using a positive decreasing sequence $\{u_n\}$ (we determine later), let

$$\Xi_n = \{0\} \cup \{\pm u_n \mathcal{E}^{(l,m)} : l, m \in \{1, 2, \cdots, K\}\}.$$

Here, Ξ_n for each n consists of matrices with single non-zero value u_n or $-u_n$ and the zero matrix. Therefore, $|\Xi_n| = 2K^2 + 1$. We define the maximum norm of matrices defined as

$$\|V\|_M = \max_{i,j} |V_{ij}|.$$

Note that for all $\theta \in \Theta$ and for all x^n, we can select $\xi \in \Xi_n$ so that

$$\xi \cdot V(\theta; x^n) = u_n \|V(\theta; x^n)\|_M. \tag{2.25}$$

Actually, when the (i, j) element of $V(\theta; x^n)$ satisfies $|V_{ij}(\theta; x^n)| = \|V(\theta; x^n)\|_M$, either of $\pm u_n \mathcal{E}^{(i,j)} \in \Xi_n$ with the same signature as the element achieves (2.25). Note that by definition of the maximum norm, such index (i, j) always exists for $V(\theta; x^n) \neq 0$.

Since $|\Xi_n| = 2K^2 + 1$, to encode $\xi \in \Xi_n$, we could use the uniform code length $\log(2K^2 + 1)$. However, we employ a variable length code to obtain a smaller regret bound. Let $l_2(n) = n^{-\nu}$ $(\nu > 0)$ and

$$\bar{l}_2(n) = \log \frac{1}{1 - e^{-l_2(n)}} + \log(2K^2).$$

We design

$$\tilde{L}_n(\xi) = \begin{cases} l_2(n), & \text{if } \xi = 0, \\ \bar{l}_2(n), & \text{otherwise.} \end{cases}$$

This $L_n(\xi)$ satisfies Kraft's inequality on Ξ_n as the equality. Now, we have defined a new two part code $(\Theta \times \Xi_n, \alpha\bar{L}_n)$ with a hyper parameter u. Note that

$$\bar{l}_2(n) \leq l_2(n) - \log l_2(n) + \log(2K^2) \tag{2.26}$$

holds, which follows from the inequality $e^{-x} \leq 1 - e^{-x}x$ equivalent to $e^x \geq 1 + x$.

We show that by choosing the value of u_n properly, the two part code has a good regret bound. The proper value is determined based on some assumptions on $\mathcal{M}$.

2.5.3 Regret Bounds for the New Two-Part Codes

To obtain an upper bound of the regret of the new two part code, we need some preliminaries and assumptions. Let $\delta_n = (g \log n/n)^{1/2}$ $(g > 0)$. We define two subsets of $\mathcal{X}^n$ as follows:

$$\mathcal{G}_n = \{x^n : \|V(\hat{\theta}(x^n); x^n)\|_M \leq \delta_n, \hat{\theta}(x^n) \in \Theta^\circ\},$$
$$\mathcal{G}_n^c = \{x^n : \|V(\hat{\theta}(x^n); x^n)\|_M > \delta_n, \hat{\theta}(x^n) \in \Theta^\circ\}.$$

We will show that our two part code based on $(\ddot{\Theta}_n \times \Xi_n, \alpha\bar{L}_n)$ has the regret bound similar to (2.21) for $x^n \in \mathcal{G}_n$ and smaller regret bound for $x^n \in \mathcal{G}_n^c$ and large n. Consequently, the two part code has the regret bound similar to the bound for exponential family cases (2.21) for $x^n \in \mathcal{G}_n \cup \mathcal{G}_n^c$ (i.e. x^n with $\hat{\theta}(x^n) \in \Theta^\circ$) when n is sufficiently large.

We need additional assumptions on $\mathcal{M}$.

Assumption 7. *There exist $\kappa' > 0$ and $\bar{b}' > 0$ such that the following holds, for all $x^n \in \mathcal{G}_n$, for all $\theta : |\theta - \hat{\theta}| \leq \bar{b}'$, and for all $z \in \mathbb{R}^K$.*

$$z^T \hat{J}(\theta, x^n) z \leq (1 + \kappa'|\theta - \hat{\theta}|) z^T \hat{J}(\hat{\theta}, x^n) z.$$

This Assumption 7 is used for $x^n \in \mathcal{G}_n$, while for $x^n \in \mathcal{G}_n^c$, we use the following.

Assumption 8. *There exist $\epsilon > 0$ and $C_\epsilon > 0$ such that the following holds for all $x^n \in \mathcal{G}_n^c$ and for all $z \in \mathbb{R}^K$,*

$$\max_{\theta:|\theta-\hat{\theta}|\le\epsilon} z^T \hat{J}(\theta; x^n) z \le C_\epsilon z^T \Big(\hat{J}(\hat{\theta}; x^n) + J(\hat{\theta}) \Big) z.$$

Assumption 9. *There exist certain constants $\Delta > 0$ and $\gamma \in (0,1)$ such that for all $x^n \in \mathcal{G}_n^c$,*

$$\inf_{\theta:|\theta-\hat{\theta}|\le\Delta} \frac{||V(\theta; x^n)||_M}{||V(\hat{\theta}; x^n)||_M} > \gamma.$$

Note that for all $x^n \in \mathcal{G}_n^c$ and for all θ such that $|\theta - \hat{\theta}| \le \Delta$, we have $||V(\theta; x^n)||_M > \gamma\delta_n$.

Assumption 10. *There exist certain constants $\bar{\delta} > 0$ and $B > 0$ such that, for all i, j, k, l,*

$$\sup_{\theta\in\Theta} \sup_{\xi:||\xi||_M\le\bar{\delta}} \left| \mathrm{E}_{\bar{p}_{\theta,\xi}}[V_{ij}(\theta; X) V_{kl}(\theta; X)] \right| \le B.$$

The following theorem gives an upper bound of the code length of our two part code and suggests the value of the hyper parameter u_n as $\gamma\delta_n/B$.

Theorem 2.4. *Under Assumptions 1–5 and 8–10, the two part code based on $(\ddot{\Theta}_n \times \Xi_n, \alpha\bar{L}_n)$ with $u_n = \gamma\delta_n/B$ and $\gamma g/2B - \nu\alpha > 0$ has the following regret bound for all n satisfying*

$$n > \max\Big\{ \frac{Ka^2}{4\zeta} \max\Big\{ \frac{1}{\epsilon^2}, \frac{1}{\Delta^2} \Big\}, \exp\Big(\frac{C_\epsilon K a^2 + \alpha \log(2K^2)}{\gamma g/2B - \nu\alpha} \Big), \frac{1}{\kappa^4 K^2 a^4}, \frac{4\zeta}{\kappa^2 K a^2} \Big\}, \tag{2.27}$$

$$\sup_{x^n:\hat{\theta}\in\Theta^\circ} \frac{1}{\alpha} \mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \le \frac{K}{2} \log \frac{n}{2\pi} + \log \int_\Theta |J(\theta)|^{1/2} d\theta + f_{\mathrm{ne}}(n), \tag{2.28}$$

where

$$\begin{aligned} f_{\mathrm{ne}}(n) &= \frac{K}{2} \log 2\pi - K \log a + \frac{C_{\mathcal{G}_n,n} K a^2}{8\alpha} (1 + K\delta_n) + r(n) + l_2(n), \\ C_{\mathcal{G}_n,n} &= C_n (1 + \kappa' K^{1/2} a \zeta^{-1/2} n^{-1/2}/2), \\ r(n) &= \log(1 + C_J n^{-1/4}) + \log(1 + C_\Theta n^{-1/4}) + \log(1 + C_{J,K} n^{-1/4}). \end{aligned} \tag{2.29}$$

Remark: Note that $\lim_{n\to\infty} C_{\mathcal{G}_n,n} = 1$ and $\lim_{n\to\infty} r(n) = 0$. (Recall that C_n is given as (2.18) and $C_n = 1 + O(n^{-1/4})$ for $\beta = 1/4$.) This result corresponds to (2.21) for exponential families.

Under Assumption 6, we can construct $p'^{(n)}_{\mathrm{tp}}$ in (2.22) with $p^{(n)}_{\mathrm{tp}} = \bar{p}^{(n)}_{\mathrm{tp}}$. Let

$$\overline{\mathrm{REG}}(n) = \alpha\Big(\frac{K}{2}\log\frac{n}{2\pi} + \log\int_{\Theta}|J(\theta)|^{1/2}d\theta + f_{\mathrm{ne}}(n) + l_1(n)\Big).$$

Then, we have the following theorem.

Theorem 2.5 (Bounds for non-exponential families)**.** *Let $\mathcal{M}$ is a non-exponential family which satisfies Assumptions 1–10. When $p^* \in \mathcal{M}$, the MDL estimator induced by $p'^{(n)}_{\mathrm{tp}}$ satisfies the following inequalities for large n.*

$$\mathrm{E}_{X^n\sim p^*}\bar{d}_\lambda(p^*\|\ddot{p}_{X^n}) \le \frac{1}{n}\overline{\mathrm{REG}}(n).$$

and for all $b > 0$,

$$\Pr\Big\{\bar{d}_\lambda(p^*\|\ddot{p}_{X^n}) > \frac{1}{n}\overline{\mathrm{REG}}(n) + b\Big\} < e^{-n\alpha^{-1}b},$$

where Pr *is measured with $X^n \sim p^*$.*

We have obtained the bounds for non-exponential families which generalize the bounds for exponential families given by Theorem 2.3.

In the rest of this section, we will prove Theorem 2.4. We need some definitions and lemmas to prove the theorem.

Let $\|V\|_s$ denote the spectral norm of a real-symmetric matrix V:

$$\|V\|_s = \max_{|z|=1}|z^T V z|.$$

The following lemma provides inequalities between the maximum norm and the spectral norm.

Lemma 5. *For an arbitrary $K \times K$ real-symmetric matrix V, the following inequality holds.*

$$\frac{1}{\sqrt{K}}\|V\|_M \le \|V\|_s \le K\|V\|_M. \tag{2.30}$$

Proof of Lemma 5. To prove (2.30), we use the Frobenius norm defined as

$$\|V\|_F^2 = \sum_{i,j} V_{ij}^2.$$

By definition of $\|V\|_M$, $\|V\|_F^2 \leq K^2\|V\|_M^2$ holds. Further, since $\|V\|_s^2$ is the maximum squared eigenvalue of V, and since $\|V\|_F^2 = \mathrm{Tr}(V^2)$ corresponds to the sum of the eigenvalues of V^2, which equals the sum of the squared eigenvalues of V, we have $\|V\|_s^2 \leq \|V\|_F^2$. By combining them, we have $\|V\|_s \leq K\|V\|_M$. In the same manner, $\|V\|_F^2 \leq K\|V\|_s^2$ and $\|V\|_M^2 \leq \|V\|_F^2$ holds. Therefore, we have $\|V\|_M \leq \sqrt{K}\|V\|_s$. □

We define a function $g(\theta, \xi; x^n)$, which appears in evaluation of the code length of our two part code, as follows:

$$g(\theta, \xi; x^n) = \xi \cdot V(\theta; x^n) - \psi_\theta(\xi).$$

The following lemma is used to provide an upper bound of the regret for $x^n \in \mathcal{G}_n^c$.

Lemma 6. *Let Δ be a constant Δ in Assumption 9. Suppose $x^n \in \mathcal{G}_n^c$ and $|\ddot{\hat{\theta}} - \hat{\theta}| < \Delta$. Then, under Assumptions 9 and 10, for $\xi \in \Xi_n$ satisfying* (2.25)*,*

$$g(\ddot{\hat{\theta}}, \xi; x^n) > u_n\|V(\ddot{\hat{\theta}}; x^n)\|_M\Big(1 - \frac{B}{2\gamma\delta_n}u_n\Big).$$

Proof of Lemma 6. When ξ satisfies (2.25),

$$g(\ddot{\hat{\theta}}, \xi; x^n) = u_n\|V(\ddot{\hat{\theta}}; x^n)\|_M - \psi_{\ddot{\hat{\theta}}}(\xi).$$

holds. First, we evaluate $\psi_{\ddot{\hat{\theta}}}(\xi)$. Suppose $|\xi_{ij}| = u_n$ and $\xi_{i'j'} = 0$ for $(i', j') \neq (i, j)$. By the definition of ψ_θ, we have

$$\frac{\partial\psi_\theta(\xi)}{\partial\xi_{ij}}\Big|_{\xi=0} = \mathrm{E}_{p_\theta}V_{ij}(\theta; X) = 0$$

As for the second derivative, we have for any $v \in \mathbb{R}^{K\times K}$,

$$\begin{aligned}\sum_{ijkl} v_{kl}v_{ij}\frac{\partial^2\psi_\theta(\xi)}{\partial\xi_{kl}\partial\xi_{ij}} &= \sum_{ijkl} v_{kl}v_{ij}\mathrm{E}_{\bar{p}_{\theta,\xi}}V_{kl}(\theta;X)V_{ij}(\theta;X) - \sum_{ijkl} v_{kl}v_{ij}E_{\bar{p}_{\theta,\xi}}V_{kl}(\theta;X)E_{\bar{p}_{\theta,\xi}}V_{ij}(\theta;X)\\ &= \sum_{ijkl} v_{kl}v_{ij}\mathrm{E}_{\bar{p}_{\theta,\xi}}V_{kl}(\theta;X)V_{ij}(\theta;X) - \Big(\sum_{ij} v_{ij}E_{\bar{p}_{\theta,\xi}}V_{ij}(\theta;X)\Big)^2\\ &\le \sum_{ijkl}|v_{kl}v_{ij}|B.\end{aligned}$$

Since $\psi_\theta(0) = \log\int p_\theta(x)dx = 0$, by Taylor expansion around $\xi_{ij} = 0$, we have for all θ,

$$\psi_\theta(\xi) \le \frac{B}{2}\sum_{ijkl}|\xi_{ij}\xi_{kl}| = \frac{B}{2}u_n^2.$$

Therefore, we have

$$\begin{aligned}g(\ddot{\hat{\theta}},\xi;x^n) &\ge u_n\|V(\ddot{\hat{\theta}};x^n)\|_M - \frac{B}{2}u_n^2\\ &> u_n\|V(\ddot{\hat{\theta}};x^n)\|_M\Big(1-\frac{B}{2\gamma\delta_n}u_n\Big).\end{aligned}$$

□

Now, we are ready to show Theorem 2.4.

Proof of Theorem 2.4. Since we need the bound only for x^n such that $\hat{\theta} \in \Theta^\circ$, we suppose $\hat{\theta} \in \Theta^\circ$ in this proof. By (2.24), the code length of our code is

$$\begin{aligned}-\log\bar{p}_{\mathrm{tp}}^{(n)}(x^n) &= -\log p_{\ddot{\theta}}(x^n) - n\Big\{\ddot{\xi}\cdot V(\ddot{\theta};x^n) - \psi_{\ddot{\theta}}(\xi)\Big\} + \alpha\bar{L}_n(\ddot{\theta},\ddot{\xi})\\ &= -\log p_{\ddot{\theta}}(x^n) - ng(\ddot{\theta},\ddot{\xi};x^n) + \alpha\bar{L}_n(\ddot{\theta},\ddot{\xi})\\ &\le -\log p_{\ddot{\hat{\theta}}}(x^n) - ng(\ddot{\hat{\theta}},\bar{\xi};x^n) + \alpha\bar{L}_n(\ddot{\hat{\theta}},\bar{\xi}),\end{aligned}$$

where $\ddot{\hat{\theta}}$ is one of the closest point in $\ddot{\Theta}$ to $\hat{\theta}$ and $\bar{\xi}$ denotes the element of Ξ_n which we will define depending on whether $x^n \in \mathcal{G}_n$ or not. Hence, we have

$$\mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)};\mathcal{M},x^n) = \log\frac{p_{\hat{\theta}}(x^n)}{\bar{p}_{\mathrm{tp}}^{(n)}(x^n)} \le \log\frac{p_{\hat{\theta}}(x^n)}{p_{\ddot{\hat{\theta}}}(x^n)} - ng(\ddot{\hat{\theta}},\bar{\xi};x^n) + \alpha\bar{L}_n(\ddot{\hat{\theta}},\bar{\xi}). \tag{2.31}$$

We are to give upper bounds on the right side of (2.31).

Part I ($x^n \in \mathcal{G}_n$): First, we consider the case $x^n \in \mathcal{G}_n$, for which we let $\bar{\xi} = 0$. Then, we have

$$\mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \leq \log \frac{p_{\hat{\theta}}(x^n)}{p_{\ddot{\theta}}(x^n)} + \alpha \bar{L}_n(\ddot{\hat{\theta}}, 0).$$

By the same Taylor expansion as (2.19), we have

$$\mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \leq \frac{n}{2}(\ddot{\hat{\theta}} - \hat{\theta})^T \hat{J}(\theta'; x^n)(\ddot{\hat{\theta}} - \hat{\theta}) + \alpha \bar{L}_n(\ddot{\hat{\theta}}, 0),$$

where θ' is a point on the line segment between $\ddot{\hat{\theta}}$ and $\hat{\theta}$. Recall $|\theta' - \hat{\theta}|^2 \leq Ka^2/4n\zeta$. (See the proof of Lemma 3.) Hence by Assumption 7, we have

$$\mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \leq \frac{n}{2}(\ddot{\hat{\theta}} - \hat{\theta})^T \hat{J}(\hat{\theta}; x^n)(\ddot{\hat{\theta}} - \hat{\theta})(1 + \kappa' K^{1/2} a \zeta^{-1/2} n^{-1/2}/2) + \alpha \bar{L}_n(\ddot{\hat{\theta}}, 0), \quad (2.32)$$

To evaluate the first term of the right side, we will prove

$$z^T \hat{J}(\theta; x^n) z \leq z^T J(\theta) z \Big(1 + \|V(\theta; x^n)\|_s\Big) \quad (2.33)$$

for all $z \in \mathbb{R}^K$ and all $\theta \in \Theta$, as follows. By the definition of $V(\theta; x^n)$, for all $z \in \mathbb{R}^K$,

$$(J^{1/2}(\theta) z)^T V(\theta; x^n)(J^{1/2}(\theta) z) = z^T \hat{J}(\theta; x^n) z - z^T J(\theta) z$$

and

$$\Big|J^{1/2}(\theta) z\Big|^2 = z^T J(\theta) z$$

hold. Hence we have the following inequality for all θ and x^n.

$$z^T \hat{J}(\theta; x^n) z \leq z^T J(\theta) z \Big(1 + \|V(\theta; x^n)\|_s\Big),$$

which is (2.33).

Hence from (2.32), we have

$$\mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \le \frac{n}{2}(\ddot{\hat{\theta}} - \hat{\theta})^T J(\hat{\theta})(\ddot{\hat{\theta}} - \hat{\theta})\Big(1 + \|V(\hat{\theta}; x^n)\|_s\Big)(1 + \kappa' K^{1/2} a \zeta^{-1/2} n^{-1/2}/2) + \alpha \bar{L}_n(\ddot{\hat{\theta}}, 0).$$

Then, by Lemma 3, and since $\|V(\hat{\theta}; x^n)\|_s \le K\|V(\hat{\theta}; x^n)\|_M \le K\delta_n$ holds for $x^n \in \mathcal{G}_n$, we have

$$\begin{aligned}\max_{x^n \in \mathcal{G}_n} \mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) &\le \frac{C_n K a^2}{8}(1 + K\delta_n)(1 + \kappa' K^{1/2} a \zeta^{-1/2} n^{-1/2}/2) + \alpha \bar{L}_n(\ddot{\hat{\theta}}, 0) \\ &\le \frac{C_{\mathcal{G}_n, n} K a^2}{8}(1 + K\delta_n) + \alpha(L_n(\ddot{\hat{\theta}}) + l_2(n)), \end{aligned} \tag{2.34}$$

where $C_{\mathcal{G}_n,n}$ is defined as (2.29). Recalling $\bar{L}_n(\ddot{\hat{\theta}}, 0) = L_n(\ddot{\hat{\theta}}) + l_2(n)$ and Lemma 1, (2.34) is bounded upper by α times the right side of (2.28). That is, we have the demanded upper bound for $x^n \in \mathcal{G}_n$.

Part II ($x^n \in \mathcal{G}_n^c$): When $x^n \in \mathcal{G}_n^c$ holds, $\|V(\hat{\theta}; x^n)\|_M$ is not bounded in general. However, Lemma 6 allows us to achieve a smaller regret than the minimax value.

Let $\bar{\xi}$ be an element of Ξ_n such that (2.25) holds. Note that since (2.13) implies $|\ddot{\hat{\theta}} - \hat{\theta}|^2 \le Ka^2/4n\zeta$, the condition $|\ddot{\hat{\theta}} - \hat{\theta}| < \Delta$ in Lemma 6 is satisfied when

$$n > \frac{Ka^2}{4\zeta\Delta^2}. \tag{2.35}$$

Since we let $u_n = \gamma\delta_n/B$, for the second term of (2.31), by Lemma 6,

$$-ng(\ddot{\hat{\theta}}, \bar{\xi}; x^n) < -n\frac{\gamma\delta_n}{2B}\|V(\ddot{\hat{\theta}}; x^n)\|_M$$

holds for n satisfying (2.35).

Next, we evaluate the first term of the right side of (2.31). By Taylor expansion, we have

$$\log \frac{p_{\hat{\theta}}(x^n)}{p_{\ddot{\hat{\theta}}}(x^n)} = \frac{n(\ddot{\hat{\theta}} - \hat{\theta})^T \hat{J}(\theta', x^n)(\ddot{\hat{\theta}} - \hat{\theta})}{2},$$

where θ' is a point on the line between $\ddot{\hat{\theta}}$ and $\hat{\theta}$. Let ϵ be so small that Assumption 8 holds. Then, by Assumption 8, for all unit vector z, and for all θ such that $|\theta - \hat{\theta}| < \epsilon$, the following

holds

$$\frac{z^T \hat{J}(\theta; x^n) z}{z^T J(\hat{\theta}) z} \leq \frac{C_\epsilon \Big(z^T \hat{J}(\hat{\theta}; x^n) z + z^T J(\hat{\theta}) z \Big)}{z^T J(\hat{\theta}) z},$$

which yields

$$z^T \hat{J}(\theta; x^n) z \leq z^T J(\hat{\theta}) z C_\epsilon (1 + \|V(\hat{\theta}; x^n)\|_s).$$

Now, assume

$$n > \frac{Ka^2}{4\zeta\epsilon^2}. \tag{2.36}$$

Then, by Proposition 1, $|\ddot{\hat{\theta}} - \hat{\theta}| < \epsilon$ holds. Hence, by Lemma 3, the first term of (2.31) is bounded as

$$\log \frac{p_{\hat{\theta}}(x^n)}{p_{\ddot{\hat{\theta}}}(x^n)} \leq \frac{C_n C_\epsilon K a^2 (1 + \|V(\hat{\theta}; x^n)\|_s)}{8} \leq \frac{C_n C_\epsilon K a^2 (1 + K \|V(\hat{\theta}; x^n)\|_M)}{8}$$

Therefore, together with the bound for the second term, we have

$$\log \frac{p_{\hat{\theta}}(x^n)}{\bar{p}_{\ddot{\hat{\theta}}, \bar{\xi}}(x^n)} < \frac{C_n C_\epsilon K a^2}{8} - \|V(\hat{\theta}; x^n)\|_M \Big(\frac{\gamma \delta_n}{2B} \frac{\|V(\ddot{\hat{\theta}}; x^n)\|_M}{\|V(\hat{\theta}; x^n)\|_M} n - \frac{C_n C_\epsilon K a^2}{8} \Big).$$

Under Assumption 9 and (2.35), we have

$$\log \frac{p_{\hat{\theta}}(x^n)}{\bar{p}_{\ddot{\hat{\theta}}, \bar{\xi}}(x^n)} < \frac{C_n C_\epsilon K a^2}{8} - \|V(\hat{\theta}; x^n)\|_M \Big(\frac{\gamma \delta_n}{2B} n - \frac{C_n C_\epsilon K a^2}{8} \Big).$$

When n satisfies

$$n > \frac{C_n C_\epsilon K a^2}{4\gamma\delta_n}, \tag{2.37}$$

the second term is negative. Then when n satisfies (2.37), for all $x^n \in \mathcal{G}_n^c$, since $\|V(\hat{\theta}; x^n)\| > \delta_n$, we have

$$\log \frac{p_{\hat{\theta}}(x^n)}{\bar{p}_{\ddot{\hat{\theta}}, \bar{\xi}}(x^n)} < \frac{C_n C_\epsilon K a^2}{8} - \frac{\gamma \delta_n^2}{2B} n + \frac{C_n C_\epsilon K a^2}{8} \delta_n.$$

Now, when n satisfies (2.36) and (2.35), which are denoted by

$$n > \frac{Ka^2}{4\zeta} \max\Big\{\frac{1}{\epsilon^2}, \frac{1}{\Delta^2}\Big\},$$

and satisfies (2.37), we have the following upper bound of the regret.

$$\max_{x^n \in \mathcal{G}_n^c} \mathrm{REG}(\bar{p}_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) < \frac{C_n C_\epsilon K a^2 (1+\delta_n)}{8} - \frac{\gamma \delta_n^2}{2B} n + \alpha(L_n(\ddot{\hat{\theta}}) + \bar{l}_2(n)). \tag{2.38}$$

Lastly, we will give the condition under which the bound (2.38) is smaller than (2.34). The right side of (2.34) minus that of (2.38) is

$$\begin{aligned}
&\frac{C_{\mathcal{G}_n,n} K a^2 (1+K\delta_n)}{8} + \alpha(L_n(\ddot{\hat{\theta}}) + l_2(n)) - \frac{C_n C_\epsilon K a^2 (1+\delta_n)}{8} + \frac{\gamma \delta_n^2}{2B} n - \alpha(L_n(\ddot{\hat{\theta}}) + \bar{l}_2(n)) \\
&> -\frac{C_n C_\epsilon K a^2 (1+\delta_n)}{8} + \alpha(l_2(n) - \bar{l}_2(n)) + \frac{\gamma \delta_n^2}{2B} n \\
&> \frac{\gamma \delta_n^2}{2B} n - \frac{C_n C_\epsilon K a^2 (1+\delta_n)}{8} + \alpha(\log l_2(n) - \log(2K^2)) \\
&> \frac{\gamma \delta_n^2}{2B} n - C_\epsilon K a^2 + \alpha(\log l_2(n) - \log(2K^2)) \\
&= \frac{\gamma g \log n}{2B} - C_\epsilon K a^2 - \nu\alpha \log n - \alpha \log(2K^2) \\
&= \Big(\frac{\gamma g}{2B} - \nu\alpha\Big) \log n - C_\epsilon K a^2 - \alpha \log(2K^2),
\end{aligned}$$

where the second inequality follows from (2.26) and we assume n satisfies $\delta_n \le 1$ and $C_n \le 4$. Since $\gamma g/2B - \nu\alpha > 0$ is assumed, the above difference is positive, if

$$\log n > \frac{C_\epsilon K a^2 + \alpha \log(2K^2)}{\gamma g/2B - \nu\alpha}$$

holds. Finally, we consider sufficient conditions for $\delta_n \le 1$ and $C_n \le 4$, respectively. Since $x \ge \log 2x$ holds for $x \ge 1$, we have $\log g \ge \log(2 \log g) = \log 2 + \log\log g$ for $g \ge e$. Further, $\log n/n$ decreases when $n \ge e$. Therefore, if $n \ge 2g \log g$, we have

$$\delta_n^2 = \frac{g \log n}{n} \le \frac{\log\log g + \log(2g)}{2 \log g} \le \frac{2 \log g}{2 \log g} = 1.$$

As for C_n, since we assume $\beta = 1/4$ here, when

$$n \geq \max\Big\{\frac{1}{\kappa^4 K^2 a^4}, \frac{4\zeta}{\kappa^2 K a^2}\Big\}.$$

Therefore, (2.27) is a sufficient condition for (2.28). We have obtained the claim of the theorem. □

2.6 Proofs of a Proposition and Lemmas

Proof for Proposition 1. Recall that the quantization is realized by the hyper rectangles which circumscribe the ellipsoids defined as (2.12). Note that the ellipsoid of scale $\sqrt{K}$ circumscribes the rectangle which circumscribes the original ellipsoid.

In a hyper rectangle induced by an ellipsoid (2.12), the points farthest from the center of mass θ of the hyper rectangle are on its corners. Let θ' be one of these corners. Since $|\theta' - \theta|$ is not greater than $\sqrt{K}$ times of the largest length of the edges of the rectangle, all the corners are inside of the following ellipsoid with $\sqrt{K}$ times larger scale as

$$\Big\{\theta' : (\theta' - \theta)^T J(\theta_{\Im})(\theta' - \theta) \leq \frac{K a^2}{4n}\Big\}.$$

Therefore, (2.13) holds. Finally, (2.14) can easily be derived as well. □

Proof for Lemma 1. For each large cell $\Im$, let $|\ddot{\Theta}_{n,\Im}|$ denote the number of the small cells which intersect $\Im$. We will evaluate its upper bound. For a measurable bounded subset $E \subset \mathbb{R}^K$, we denote the volume of E as

$$\sigma(E) = \int_E d\theta.$$

Let $\bar{\Im}$ denote the union of all the hyper rectangles which intersect $\Im$, and σ_{hr} the volume of the hyper rectangles. Note that $\Im \subseteq \bar{\Im}$ and that $\sigma_{\mathrm{hr}} = (an^{-1/2})^K |J(\theta_{\Im})|^{-1/2}$. The number of

small cells which intersect $\mathfrak{S}$ is given by

$$|\ddot{\Theta}_{n,\mathfrak{S}}| = \frac{\sigma(\bar{\mathfrak{S}})}{\sigma_{\rm hr}}. \tag{2.39}$$

We will give an upper bound of $\sigma(\bar{\mathfrak{S}})$. Let $N_{\mathfrak{S}}$ be the number of the small cells which intersect the surface of $\mathfrak{S}$. First, we evaluate an upper bound of $N_{\mathfrak{S}}$. Recall that $\mathfrak{S}$ is the intersection of the convex set Θ and a hyper cube (with side lengths $an^{-\beta}$), which we will denote by $C(\mathfrak{S})$ here. Note that the number of the hyper rectangles to cover the surface of $C(\mathfrak{S})$ is larger than $N_{\mathfrak{S}}$. When the number is maximized, the smallest edge of the rectangle is parallel to an edge of the hyper cube $C(\mathfrak{S})$. Further the number is upper bounded by that for the case in which the lengths of all the edges are equal to the minimum value. Since, from Assumption 2, the maximum of the eigenvalue of $J(\theta_{\mathfrak{S}})$ is at most $\bar{\lambda}$, the length of the shortest edge of the rectangle is at least $an^{-1/2}\bar{\lambda}^{-1/2}$. Then, we have

$$\begin{aligned} N_{\mathfrak{S}} &\le 2K\Big(\frac{an^{-\beta}}{an^{-1/2}\bar{\lambda}^{-1/2}} + 2\Big)^{K-1} \\ &\le 2Kn^{(K-1)(1/2-\beta)}(\bar{\lambda}^{1/2} + 2)^{K-1}, \end{aligned}$$

where the factor $2K$ is the number of the $K-1$ dimensional hyper cubes which form the surface of $C(\mathfrak{S})$. Let $\bar{N}$, which is independent of $\mathfrak{S}$, denote the last side of the above inequalities. Then, we have

$$|\ddot{\Theta}_{n,\mathfrak{S}}| = \frac{\sigma(\bar{\mathfrak{S}})}{\sigma_{\rm hr}} \le \frac{\sigma(\mathfrak{S})}{\sigma_{\rm hr}} + \bar{N}.$$

Further, recall that $\sigma_{\rm hr} = (an^{-1/2})^K |J(\theta_{\mathfrak{S}})|^{-1/2}$. Then, we have

$$\begin{aligned} \sigma(\bar{\mathfrak{S}}) &\le \sigma(\mathfrak{S}) + \bar{N}\sigma_{\rm hr} \\ &\le \sigma(\mathfrak{S}) + \bar{N}(an^{-1/2})^K |J(\theta_{\mathfrak{S}})|^{-1/2} \\ &\le \sigma(\mathfrak{S}) + 2Kn^{(K-1)(1/2-\beta)}(\bar{\lambda}^{1/2} + 2)^{K-1}(an^{-1/2})^K |J(\theta_{\mathfrak{S}})|^{-1/2} \\ &\le \sigma(\mathfrak{S}) + a^K C_J n^{-K\beta-(1/2-\beta)}, \end{aligned}$$

where we let

$$Z = \min_{\theta \in \Theta} |J(\theta)|$$

and

$$C_J = 2K(\bar{\lambda}^{1/2} + 2)^{K-1} Z^{-1/2}.$$

Under Assumptions 2, $Z^{-1/2} \leq \zeta^{-K/2}$ is finite.

Since $\Im$ is a subset of the hyper cube with side lengths $an^{-\beta}$, which induces it, $\sigma(\Im) \leq (an^{-\beta})^K$ holds. Then,

$$\sigma(\bar{\Im}) \leq (an^{-\beta})^K (1 + C_J n^{-(1/2-\beta)}) \tag{2.40}$$

holds.

When $\Im \cap \partial\Theta = \emptyset$, since $\Im$ is a hyper cube, $\sigma(\Im) = (an^{-\beta})^K$ holds. We have

$$\sigma(\bar{\Im}) \leq \sigma(\Im)(1 + C_J n^{-(1/2-\beta)}). \tag{2.41}$$

Now, we can derive the following inequality under Assumption 4.

$$\sigma(\Im) \leq \frac{\int_{\Im} |J(\theta)|^{1/2} d\theta}{|J(\theta_{\Im})|^{1/2}} (1 + C_\Theta n^{-\beta}), \tag{2.42}$$

where

$$C_\Theta = \max_{\theta \in \Theta} \frac{KaD_J}{|J(\theta)|^{1/2}}.$$

In fact, for all $\theta \in \Im$, by Taylor expansion, we have

$$\begin{aligned} |J(\theta_{\Im})|^{1/2} &\leq |J(\theta)|^{1/2} + KaD_J n^{-\beta} \\ &= |J(\theta)|^{1/2} \Big(1 + \frac{|J(\theta)|^{1/2}}{KaD_J} n^{-\beta}\Big) \\ &\leq |J(\theta)|^{1/2} (1 + C_\Theta n^{-\beta}) \end{aligned}$$

That is,

$$1 \leq \frac{|J(\theta)|^{1/2}}{|J(\theta_{\Xi})|^{1/2}}(1 + C_{\Theta} n^{-\beta}).$$

By integrating both sides, we have (2.42).

Finally, by using (2.39), (2.41) and (2.42), we have for Ξ such that $\Xi \cap \partial\Theta = \emptyset$,

$$|\ddot{\Theta}_{n,\Xi}| \leq a^{-K} n^{K/2}(1 + C_J n^{-(1/2-\beta)})(1 + C_{\Theta} n^{-\beta}) \int_{\Xi} |J(\theta)|^{1/2} d\theta.$$

Let Θ_{∂} denote the union of large cells which have intersections with $\partial\Theta$. By taking the summation over all large cells without intersections with $\partial\Theta$, an upper bound of the number of small cells induced by such large cells is evaluated as

$$\begin{aligned} &a^{-K} n^{K/2}(1 + C_J n^{-(1/2-\beta)})(1 + C_{\Theta} n^{-\beta}) \int_{\Theta \setminus \Theta_{\partial}} |J(\theta)|^{1/2} d\theta \\ &\leq a^{-K} n^{K/2}(1 + C_J n^{-(1/2-\beta)})(1 + C_{\Theta} n^{-\beta}) \int_{\Theta} |J(\theta)|^{1/2} d\theta. \end{aligned} \tag{2.43}$$

Next, we consider Ξ (large cell) with $\Xi \cap \partial\Theta \neq \emptyset$. Define a real number W_{Θ} as

$$W_{\Theta} = \max_i (\max\{\theta_i : \theta \in \Theta\} - \min\{\theta_i : \theta \in \Theta\}).$$

Since Θ is included in the hyper cube with side W_{Θ}, the number of large cells (hyper cubes) which have intersections with $\partial\Theta$ is at most

$$2^K (W_{\Theta} a^{-1} n^{\beta} + 2)^{K-1} \leq C_K a^{-(K-1)} n^{(K-1)\beta}, \tag{2.44}$$

where

$$C_K = 2^K (W_{\Theta} + 2a)^{K-1}.$$

By using (2.39) and (2.40), we have for all large cells,

$$|\ddot{\Theta}_{n,\Xi}| \leq (1 + C_J n^{-(1/2-\beta)}) \Lambda^{1/2} n^{K(1/2-\beta)},$$

where

$$\Lambda = \max_{\theta \in \Theta} |J(\theta)|. \tag{2.45}$$

From Assumption 2, $\Lambda \leq \bar{\lambda}^K$ is finite. By taking the summation of $|\ddot{\Theta}_{n,\Xi}|$ over the hyper cubes which constitutes Θ_∂, since the number of them is at most the right side of (2.44), the number of small cells induced by them is at most

$$C_K \Lambda^{1/2} a^{-(K-1)} n^{K/2-\beta} (1 + C_J n^{-(1/2-\beta)}).$$

By adding this to (2.43), since $1 + C_J n^{-(1/2-\beta)}$ and $1 + C_\Theta n^{-\beta}$ are greater than 1, we have

$$|\ddot{\Theta}_n| \leq a^{-K} n^{K/2} e^{r(n)} \int_\Theta |J(\theta)|^{1/2} d\theta,$$

where

$$r(n) = \log(1+C_J n^{-(1/2-\beta)}) + \log(1+C_\Theta n^{-\beta}) + \log(1 + C_{J,K} n^{-\beta}) \tag{2.46}$$

and

$$C_{J,K} = \frac{C_K \Lambda^{1/2} a}{\int_\Theta |J(\theta)|^{1/2} d\theta}. \tag{2.47}$$

□

2.7 Application to Mixture Families

In this section, we will apply our result to mixture families which are an example of non-exponential families.

For $i = 0, 1, 2, \cdots, K$, let q_i be a probability density function on $\mathcal{X}$. The mixture family which consists of $\{q_i\}$ is the family of convex combinations of $\{q_i\}$ as follows. For $\tau \in [0, 1/(K+$

1)], we define the mixture family $\mathcal{M}_\tau$ by

$$p_\theta(x) = \sum_{i=0}^{K} \theta_i q_i(x),$$

$$\Theta = \Theta_\tau = \{\theta \in [0,1]^K : \theta_i \geq \tau \text{ for each } i \in \{0, 1, \cdots, K\}\},$$

$$\mathcal{M} = \mathcal{M}_\tau = \{p_\theta : \theta \in \Theta_\tau\},$$

where $\theta = (\theta_1, \ldots, \theta_K)$ and $\theta_0 = 1 - \sum_{i=1}^{K} \theta_i$. In this paper, we fix $\tau > 0$. Note that the naturally determined parameter space $\tilde{\Theta}$ equals Θ_τ with $\tau = 0$. In this paper, we consider the case that $\tau > 0$. Note that the set $\{\theta \in \Theta : \theta_0 = \tau\}$ is a subset of $\partial\Theta$.

Further, we assume that each q_i cannot be represented as a convex combination of the others, that is for all $i \geq 0$,

$$\min_{(\alpha_0,\alpha_1,\ldots,\alpha_K)\in[0,1]^{K+1}:\sum_{j\neq i}\alpha_j=1} D\Big(q_i \Big\| \sum_{j\neq i} \alpha_j q_j\Big) > 0. \tag{2.48}$$

This condition guarantees that the Fisher information is positive definite.

We will show some properties of the mixture family. The empirical Fisher information of the mixture family is given as follows.

$$\hat{J}_{ij}(\theta; x) = \frac{(q_i(x) - q_0(x))(q_j(x) - q_0(x))}{p_\theta^2(x)}. \tag{2.49}$$

The following Lemma holds for mixture families in general.

Lemma 7. *The empirical Fisher information of the mixture family is positive semi-definite. That is, for all unit vectors $z \in \mathbb{R}^K \setminus \{0\}$ and for all x,*

$$z^T \hat{J}(\theta; x) z \geq 0.$$

Proof. From (2.49), we have

$$\begin{aligned} z^T \hat{J}(\theta; x) z &= \sum_{i,j} \Big(\frac{q_i(x) - q_0(x)}{p_\theta(x)}\Big)\Big(\frac{q_j(x) - q_0(x)}{p_\theta(x)}\Big) z_i z_j \\ &= \Big(\sum_i \frac{q_i(x) - q_0(x)}{p_\theta(x)} z_i\Big)^2 \geq 0. \end{aligned} \tag{2.50}$$

□

We can show the positive-definiteness of $J(\theta)$. Therefore, the assumption of existence of ζ in Assumption 2 is valid for mixture families.

Lemma 8. *When $q_0, q_1, \cdots, q_K$ satisfy the condition* (2.48)*, for all θ, $J(\theta)$ is positive-definite.*

Proof. From Lemma 7, $z^T J(\theta) z \geq 0$ holds for all $z \neq 0$. Then, it is sufficient to show $z^T J(\theta) z \neq 0$ for all $z \neq 0$.

If $z^T J(\theta) z = 0$ for a certain $z \neq 0$, then $z^T \hat{J}(\theta; x) z = 0$ must hold almost surely with respect to p_θ. It imples

$$\sum_i \frac{q_i(x) - q_0(x)}{p_\theta(x)} z_i = 0$$

because of (2.50). Hence

$$\sum_{j=1}^{K} (q_j(x) - q_0(x)) z_j = 0$$

must hold almost surely with respect to each of $q_0, q_1, \cdots, q_K$. (Recall $\theta_i > 0$ for all $i \geq 0$.) Since $z \neq 0$, there exists $i > 0$ such that $z_i \neq 0$. Let $z_0 = -\sum_{i=1}^{K} z_i$. Then

$$\sum_{i=0}^{K} z_i q_i(x) = 0$$

holds almost surely with respect to each of $q_0, q_1, \cdots, q_K$. Note that $z_j \neq 0$ for at least one j. Let b denote such j. Then, we have

$$q_b(x) = -\sum_{i \neq b} (z_i / z_b) q_i(x).$$

Since this holds almost surely with respect to q_b,

$$D(q_b \| \sum_{i \neq b} (z_i / z_b) q_i) = 0$$

holds. This contradicts the condition (2.48). Therefore, $z^T J(\theta) z \neq 0$. □

For $\theta \in \Theta_\tau$, since $\theta_i \geq \tau$ for all i, we have

$$\left|\frac{q_i(x) - q_0(x)}{p_\theta(x)}\right| \leq \frac{1}{\tau}.$$

Therefore,

$$\hat{J}_{ij}(\theta; x) \leq \frac{1}{\tau^2} \tag{2.51}$$

holds for all $\theta \in \Theta_\tau$, x, i and j. This implies that $\|J(\theta)\|_M \leq 1/\tau^2$ for all $\theta \in \Theta_\tau$. Then, from (2.30), we have for all unit vectors $z \in \mathbb{R}^K$,

$$\max_{\theta \in \Theta_\tau} z^T J(\theta) z \leq \frac{K}{\tau^2}.$$

Therefore, Assumption 2 is valid for the mixture family $\mathcal{M}_\tau$ with

$$\bar{\lambda} = \frac{K}{\tau^2}.$$

This gives an upper bound on Λ defined by (2.45),

$$\Lambda \leq \bar{\lambda}^K = \left(\frac{K}{\tau^2}\right)^K.$$

Further, since

$$\frac{\partial}{\partial \theta_h} \hat{J}_{ij}(\theta; x) = -2 \frac{q_h(x) - q_0(x)}{p_\theta(x)} \hat{J}_{ij}(\theta; x), \tag{2.52}$$

we have for all $z \in \mathbb{R}^K$,

$$\left|\frac{\partial}{\partial \theta_h} z^T \hat{J}(\theta; x) z\right| \leq \frac{2}{\tau} z^T \hat{J}(\theta; x) z. \tag{2.53}$$

From Lebesgue's dominated convergence theorem, (2.53) implies

$$\left|\frac{\partial}{\partial \theta_h} z^T J(\theta) z\right| \leq \frac{2}{\tau} z^T J(\theta) z.$$

From this, we can show the following lemma which implies that Assumption 5 holds with

$$\bar{b} = \frac{\tau}{2\sqrt{K}}$$

and

$$\kappa = \frac{2e}{\tau} \sqrt{K}. \tag{2.54}$$

.

Lemma 9. *For for all $z \neq 0$, when $\theta_1, \theta_2 \in \Theta_\tau$ satisfies $|\theta_1 - \theta_2| \leq \tau/2\sqrt{K}$,*

$$\frac{z^T J(\theta_1) z}{z^T J(\theta_2) z} \leq 1 + \frac{2e}{\tau}\sqrt{K}|\theta_1 - \theta_2| \tag{2.55}$$

holds.

Proof. Since $z^T J(\theta) z > 0$, from (2.53), we have

$$\frac{\partial}{\partial \theta_h} \log z^T J(\theta) z \leq \frac{2}{\tau}$$

for all h, $\theta \in \Theta_\tau$ and $z \neq 0$. From Taylor expansion, for all $\theta_1, \theta_2 \in \Theta_\tau$ and $z \neq 0$, there exists θ' between θ_1 and θ_2 such that

$$\log z^T J(\theta_1) z = \log z^T J(\theta_2) z + \sum_h \frac{\partial}{\partial \theta_h} \log z^T J(\theta) z \Big|_{\theta=\theta'} (\theta_1 - \theta_2)_h,$$

where $(\theta_1 - \theta_2)_h$ denotes the h-th element of $\theta_1 - \theta_2$. Then from (2.53), we have

$$\begin{aligned}\log z^T J(\theta_1) z &\leq \log z^T J(\theta_2) z + \frac{2}{\tau}\sum_h |(\theta_1 - \theta_2)_h| \\ &\leq \log z^T J(\theta_2) z + \frac{2}{\tau}\sqrt{K}|\theta_1 - \theta_2|.\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{z^T J(\theta_1) z}{z^T J(\theta_2) z} &\leq e^{2\sqrt{K}|\theta_1 - \theta_2|/\tau} \\ &\leq 1 + \frac{2\sqrt{K}|\theta_1 - \theta_2|}{\tau} e^{2\sqrt{K}|\theta_1 - \theta_2|/\tau}\end{aligned}$$

holds. Here, we have used the inequality $e^t \leq 1 + te^t$ derived from $e^{-t} \geq 1 - t$. When $|\theta_1 - \theta_2| \leq \tau/2\sqrt{K}$, the second term is not larger than $2e\sqrt{K}|\theta_1 - \theta_2|/\tau$. Hence we have the inequality (2.55). □

The equation (2.52) leads to the following Lemma, which implies that Assumptions 7 and 8 are satisfied with $\bar{b}' = \tau/2\sqrt{K} (= \bar{b})$ and $\kappa' = 2e\sqrt{K}/\tau = \kappa$.

Lemma 10. *For all z, when $\theta \in \Theta_\tau$ satisfies $|\theta - \hat{\theta}| \leq \tau/2\sqrt{K}$,*

$$z^T \hat{J}(\theta_1, x^n) z \leq \Big(1 + \frac{2e\sqrt{K}|\theta - \hat{\theta}|}{\tau}\Big) z^T \hat{J}(\hat{\theta}, x^n) z \tag{2.56}$$

holds. Further, for an arbitrary constant $\epsilon > 0$, for all x^n and for all θ such that $|\theta - \hat{\theta}| < \epsilon$,

$$C_\epsilon = e^{2\sqrt{K}\epsilon/\tau}$$

satisfies the following inequality for all unit vector $z \in \mathbb{R}^K$,

$$z^T \hat{J}(\theta; x^n) z \leq C_\epsilon z^T \hat{J}(\hat{\theta}; x^n) z. \tag{2.57}$$

Proof. From (2.52),

$$\Big|\frac{\partial}{\partial \theta_h}\Big(z^T \hat{J}(\theta; x^n) z + s\Big)\Big| \leq \frac{2}{\tau}\Big(z^T \hat{J}(\theta; x^n) z + s\Big)$$

holds for any positive number s. Since $z^T \hat{J}(\theta; x^n) z + s > 0$ for any $\theta \in \Theta_\tau$, by the same way as Lemma 9, we have

$$z^T \hat{J}(\theta; x^n) z + s \leq e^{2\sqrt{K}|\theta - \hat{\theta}|/\tau}\Big(z^T \hat{J}(\hat{\theta}; x^n) z + s\Big),$$

which is

$$z^T \hat{J}(\theta; x^n) z \leq e^{2\sqrt{K}|\theta - \hat{\theta}|/\tau} z^T \hat{J}(\hat{\theta}; x^n) z + (e^{2\sqrt{K}|\theta - \hat{\theta}|/\tau} - 1)s,$$

Since $e^{2\sqrt{K}|\theta - \hat{\theta}|/\tau}$ is bounded and s can be arbitrarily small, we have

$$z^T \hat{J}(\theta; x^n) z \leq e^{2\sqrt{K}|\theta - \hat{\theta}|/\tau} z^T \hat{J}(\hat{\theta}; x^n) z. \tag{2.58}$$

Hence, similarly as (2.54), we can obtain (2.56). Derivation of the inequality (2.57) is straightforward from (2.58). □

From Assumption 2, for all θ,

$$\|J^{-1/2}(\theta)\|_s \leq \frac{1}{\sqrt{\zeta}}$$

holds and hence, we have

$$\|J^{-1/2}(\theta)\|_M \leq \sqrt{\frac{K}{\zeta}}.$$

In addition to this, derivatives of $\hat{J}(\theta; x^n)$ and $J(\theta)$ is bounded. Then, the derivative of $V(\theta; x^n)$ is also bounded. Therefore, Assumption 9 holds. The following lemma implies that Assumption 10 is satisfied.

Lemma 11. *For mixture family $\mathcal{M}_\tau$, for all $\theta \in \Theta_\tau$ and x, the following holds.*

$$\|V(\theta; x)\|_M \leq \frac{K\sqrt{K}}{\zeta\tau^2} + 1. \tag{2.59}$$

Proof. For all unit vectors $z \in \mathbb{R}^K$,

$$\begin{aligned} z^T(J^{-1/2}(\theta)\hat{J}(\theta; x)J^{-1/2}(\theta))z &\leq \|\hat{J}(\theta; x)\|_s |J^{-1/2}(\theta)z|^2 \\ &= \|\hat{J}(\theta; x)\|_s z^T J^{-1}(\theta)z \\ &\leq \|\hat{J}(\theta; x)\|_s \|J^{-1}(\theta)\|_s \end{aligned}$$

holds. From the second inequality of (2.30) and (2.51), $\|\hat{J}(\theta; x)\|_s \leq K\|\hat{J}(\theta; x)\|_M \leq K/\tau^2$ holds. Since eigenvalues of $J^{-1}(\theta)$ are inverses of eigenvalues of $J(\theta)$, we have $\|J^{-1}(\theta)\|_s \leq 1/\zeta$. Therefore, by using the first inequality of (2.30), we have

$$\|J^{-1/2}(\theta)\hat{J}(\theta; x)J^{-1/2}(\theta)\|_M \leq \frac{K\sqrt{K}}{\zeta\tau^2}.$$

By adding the unit matrix I, we have (2.59). □

Therefore, we can apply Theorem 2.4 to the mixture family.

Theorem 2.6. *For the mixture family, we can construct a two part code for x^n with $\hat{\theta} \in \Theta^\circ$ whose regret is bounded by* (2.28).

To obtain the risk bounds as in Corollary 2, we have to care the sequences x^n with $\hat{\theta} \in \partial\Theta$, that is, we have to show the target model satisfies Assumption 6. In fact, we can construct an extra two part code in Assumption 6. When $\hat{\theta} \in \partial\Theta$, some elements of $\hat{\theta}$ equal τ. First, consider a special case that $\hat{\theta}_i$ for $i = 0, 1, 2, \ldots, K'$ are larger than τ and the others equal τ. Define

$$\tilde{\Theta}' = \{\theta \in \Theta_0 : \theta_{K'+1} = \theta_{K'+2} = \cdots = \theta_K = \tau\},$$
$$\Theta' = \{\theta \in \Theta_\tau : \theta_{K'+1} = \theta_{K'+2} = \cdots = \theta_K = \tau\},$$
$$\Theta'^{\circ} = \{\theta \in \Theta_\tau : \forall i \leq K', \theta_i > \tau \text{ and } \theta_{K'+1} = \theta_{K'+2} = \cdots = \theta_K = \tau\}.$$

Then, $\Theta'^{\circ} \subset \Theta' \subset \tilde{\Theta}'$ holds and the special case here corresponds to the case that $\hat{\theta}(x^n) \in \Theta'^{\circ}$. Note that the set of densities $\{p_\theta : \theta \in \tilde{\Theta}'\}$ is the set of all the convex combinations of

$$q'_i = (1 - \tau(K - K'))q_i + \tau \sum_{j=K'+1}^{K} q_i$$

for $i \in \{0, 1, \ldots, K'\}$, which is the mixture family based on the set of probability densities $\{q'_i : 0 \leq i \leq K'\}$. Further, $\{p_\theta : \theta \in \Theta'\}$ can be a target model for Theorem 3 with the dimension K'. Therefore, we can construct a two-part code whose regret for the set $\{x^n : \hat{\theta} \in \Theta'^{\circ}\}$ is not larger than

$$\frac{K'}{2} \log n + o(\log n).$$

In this special case, though we fix the elements of θ which has the value larger than τ, there are $C(K, K')$ cases for each $K' \in \{k : 0 \leq k \leq K - 1\}$, where $C(K, K')$ stands for a binomial coefficient. The case that $\hat{\theta} \in \partial\Theta$ is included in one of the above cases. Description of the value K' and the choice of θ's elements larger than τ require extra code length to describe which case applies, but such code length does not depend on n. Therefore, the worst case regret for the case that $\hat{\theta} \in \partial\Theta$ is

$$\frac{K - 1}{2} \log n + o(\log n).$$

Hence, Assumption 6 with $d = 1$ holds for the mixture family $\mathcal{M}_\tau$ and we have the following theorem.

Theorem 2.7. *For mixture families, the two part code constructed in this section induces an MDL estimator which satisfies the followings. For all $p^* \in \mathcal{M}$ and large n,*

$$\mathrm{E}_{X^n \sim p^*} \bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) \leq \frac{1}{n}\overline{\mathrm{REG}}(n)$$

and

$$\Pr\Big\{\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) > \frac{1}{n}\overline{\mathrm{REG}}(n) + b\Big\} < e^{-n\alpha^{-1}b}$$

hold, where

$$\frac{1}{\alpha}\overline{\mathrm{REG}}(n) = \frac{K}{2}\log n + \log \int_\Theta |J(\theta')|^{1/2} d\theta' + \frac{C_{\mathcal{G}_n,n} K a^2}{8\alpha}(1 + K\delta_n) + r(n) - K\log a + c_0 + c_n$$

and $c_0, c' > 0$ is small constants used in the coding, which we can select arbitrarily. For the definitions of $C_{\mathcal{G}_n,n}$ and $r(n)$, see (2.18)*,* (2.29) *and* (2.46)*.*

Note that $\lim_{n\to\infty} r(n) = 0$ and $\lim_{n\to\infty} C_{\mathcal{G}_n,n} = 1$ and $C_\epsilon = e^{\frac{2}{\tau}\sqrt{K}\epsilon}$.

We will finish this section with evaluation of values or upper bounds of some constants which determine the upper bounds in Theorem 2.7. We already have seen

$$\bar{\lambda} \leq \frac{K}{\tau^2}$$

and

$$\Lambda \leq \Big(\frac{K}{\tau^2}\Big)^K.$$

Then, we have an upper bound for the constant

$$C_J = (\bar{\lambda}^{1/2} + 2)^{K-1}\Lambda^{1/2} \leq \Big(\frac{\sqrt{K}}{\tau} + 2\Big)^{K-1}\Big(\frac{\sqrt{K}}{\tau}\Big)^K.$$

Further, since

$$\min\{\theta_i : \theta \in \Theta_\tau\} = \tau$$

and

$$\max\{\theta_i : \theta \in \Theta_\tau\} = 1 - K\tau$$

hold for all i, we have $W_\Theta = 1 - (K+1)\tau$. Hence, we have

$$C_K = 2^K (W_\Theta + 2a)^{K-1} = 2^K (1 - (K+1)\tau + 2a)^{K-1}.$$

We also have an upper bound on $C_{J,K}$, defined as (2.47):

$$C_{J,K} \leq \Big(\frac{\sqrt{K}}{\tau}\Big)^K \frac{C_K a}{\int_\Theta |J(\theta)|^{1/2} d\theta}.$$

The constant D_J in Assumption 4, which determines C_Θ, depends on $\{q_i\}$. We cannot evaluate it generally. If we have an evaluation of D_J for given $\{q_i\}$, we can determine C_Θ and then $r(n)$.

2.8 Concluding Remarks

We have developed a way of construction of two part codes which have regret bounds asymptotically close to the minimax regret for non-exponential families under Assumptions 1–10. The regret bounds are given by Theorem 2.4. The regret bounds lead to the risk bounds of MDL estimators induced by the two part codes, which are given by Corollary 2.5. Our two part codes and MDL estimators allow us to estimate probability densities in the non-exponential families which satisfy Assumptions 1–10, under guarantee of the risk. Further, we have shown that mixture families satisfy the assumptions we need and have constructed such codes for mixture families. To examine other families is a future work.

2.9 Proof of Theorems 1 and 2

This section provides a proof of Theorem 1 and 2. Both are the well known results which imply that a good two part code defines a good MDL estimator.

2.9.1 Proof of Theorem 1

Since $\bar{d}_\lambda(p\|q)$ is increasing in $\lambda \in (0,1)$, the proof for the case that $\lambda = 1-\alpha^{-1}$ is sufficient. Hence we assume it. We have

$$\begin{aligned}
\mathrm{E}\Big(\bar{d}_\lambda(p^*\|\ddot{p}) - \frac{1}{n}\log\frac{p^*(X^n)}{p^{(n)}_{\mathrm{tp}}(X^n)}\Big) =& \mathrm{E}\Big(\bar{d}_\lambda(p^*\|\ddot{p}) - \frac{1}{n}\log\frac{p^*(X^n)}{\ddot{p}(X^n)} - \frac{\alpha L_n(\ddot{p})}{n}\Big) \\
=& \frac{\alpha}{n}\mathrm{E}\Big(\frac{n\bar{d}_\lambda(p^*\|\ddot{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\ddot{p}(X^n)} - L_n(\ddot{p})\Big) \\
=& \frac{\alpha}{n}\mathrm{E}\log\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\ddot{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\ddot{p}(X^n)} - L_n(\ddot{p})\Big) \\
\le& \frac{\alpha}{n}\log\mathrm{E}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\ddot{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\ddot{p}(X^n)} - L_n(\ddot{p})\Big) \\
\le& \frac{\alpha}{n}\log\mathrm{E}\sum_{\bar{p}\in\ddot{\mathcal{P}}}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\bar{p}(X^n)} - L_n(\bar{p})\Big) \\
=& \frac{\alpha}{n}\log\sum_{\bar{p}\in\ddot{\mathcal{P}}}\mathrm{E}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\bar{p}(X^n)} - L_n(\bar{p})\Big).
\end{aligned} \tag{2.60}$$

The first inequality follows from Jensen's inequality. The summation in the last line is

$$\sum_{\bar{p}\in\ddot{\mathcal{P}}}\mathrm{E}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(x^n)}{\bar{p}(x^n)} - L_n(\bar{p})\Big) = \sum_{\bar{p}\in\ddot{\mathcal{P}}}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha}\Big)e^{-L_n(\bar{p})}\mathrm{E}\Big(\frac{\bar{p}(x^n)}{p^*(x^n)}\Big)^{1/\alpha},$$

where

$$\mathrm{E}\Big(\frac{\bar{p}(X^n)}{p^*(X^n)}\Big)^{1/\alpha} = \mathrm{E}\Big(\prod_t\frac{\bar{p}(X_t)}{p^*(X_t)}\Big)^{1/\alpha} = \prod_t\mathrm{E}\Big(\frac{\bar{p}(X_t)}{p^*(X_t)}\Big)^{1-\lambda} = \mathrm{E}\Big(\frac{\bar{p}(X)}{p^*(X)}\Big)^{n(1-\lambda)}.$$

Further, recalling the definition of Rényi divergence

$$\begin{aligned}
\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha}\Big) &= \exp(n(1-\lambda)\bar{d}_\lambda(p^*\|\bar{p})) \\
&= \exp\Big(n(1-\lambda)\Big(-\frac{1}{1-\lambda}\log\mathrm{E}\Big(\frac{\bar{p}(X)}{p^*(X)}\Big)\Big)^{1-\lambda}\Big)\Big) \\
&= \mathrm{E}\Big(\frac{\bar{p}(X)}{p^*(X)}\Big)^{-n(1-\lambda)}.
\end{aligned}$$

Hence

$$\sum_{\bar{p}\in\ddot{\mathcal{P}}}\mathrm{E}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\bar{p}(X^n)} - L_n(\bar{p})\Big) = \sum_{\bar{p}\in\ddot{\mathcal{P}}}e^{-L_n(\bar{p})} \le 1, \tag{2.61}$$

which implies

$$\mathrm{E}\Big(\bar{d}_\lambda(p^*\|\ddot{p}) - \frac{1}{n}\log\frac{p^*(X^n)}{p_{\mathrm{tp}}^{(n)}(X^n)}\Big) \le 0.$$

This is the first inequality in the theorem. The second inequality is derived as follows

$$\begin{aligned}
\frac{1}{n}\mathrm{RED}^{(n)}(p^*, p_{\mathrm{tp}}^{(n)}) &= \frac{1}{n}\mathrm{E}\Big(\log\frac{p^*(X^n)}{\ddot{p}(X^n)e^{-\alpha L_n(\ddot{p})}}\Big) \\
&= \frac{1}{n}\mathrm{E}\min_{\bar{p}\in\ddot{\mathcal{P}}}\Big(\log\frac{p^*(X^n)}{\bar{p}(X^n)e^{-\alpha L_n(\bar{p})}}\Big) \\
&\le \frac{1}{n}\min_{\bar{p}\in\ddot{\mathcal{P}}}\mathrm{E}\Big(\log\frac{p^*(X^n)}{\bar{p}(X^n)e^{-\alpha L_n(\bar{p})}}\Big) \\
&= \frac{1}{n}\min_{\bar{p}\in\ddot{\mathcal{P}}}(nD(p^*\|\bar{p}) + \alpha L_n(\bar{p})) \\
&= \min_{\bar{p}\in\ddot{\mathcal{P}}}\Big(D(p^*\|\bar{p}) + \frac{\alpha L_n(\bar{p})}{n}\Big).
\end{aligned}$$

2.9.2 Proof of Theorem 2

Note that (2.61) implies that the expectation in the fifth line of (2.60) is not greater than 1. Hence, by Markov's inequality, we have

$$\Pr\Big\{\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\ddot{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\ddot{p}(X^n)} - L_n(\ddot{p})\Big) > e^{nb\alpha^{-1}}\Big\} \le e^{-n\alpha^{-1}b},$$

which is

$$\Pr\Big\{\bar{d}_\lambda(p^*\|\ddot{p}) > \frac{1}{n}\log\frac{p^*(X^n)}{p_{\mathrm{tp}}^{(n)}(X^n)} + b\Big\} < e^{-n\alpha^{-1}b}.$$

Chapter 3

MDL Estimators for Contaminated Gaussian Location Families

In this chapter, we consider an application of MDL estimators developed in the last chapter to an example of non-exponential families, contaminated Gaussian location families.

3.1 Introduction

We have obtained MDL estimators which enjoys a good risk bound for non-exponential families under certain assumptions by using the fiber bundles of local exponential families. We also have seen that the estimators can be applied to mixture families. In this chapter, we consider the contamination models [19] as another example of non-exponential families.

The contamination model we consider here is defined as

$$p_\theta = (1 - \nu_c)q_\theta + \nu_c g_\theta,$$

where q_θ is a model of regular data and g_θ is a model of noise. In particular, we assume that q_θ is a 1 dimensional Gaussian location family $\mathcal{N}(\theta, 1)$, while g_θ is also a Gaussian distribution with mean θ and variance larger than 1. This simple model of contamination is noted in [20] as the first example and was considered by Tukey [21]. We call such models as contaminated Gaussian location families.

It is difficult to show that the assumptions required for our results in the previous chapter strictly holds for the contamination models. Instead of doing so, we show that when given data belongs to a certain typical set, we can obtain the required properties for nice performance guarantee for MDL estimators. The idea to use a typical set for such a purpose was introduced by [22], [23] to extend the result of [2] to supervised learning. Actually, we use the following theorem, which is a conditional version of Theorems 2.1 and 2.2.

Theorem 3.1. *Let A_n be any event with probability $P_n > 0$. For all $\lambda \in (0, 1 - \alpha^{-1}]$ and $b > 0$, the following hold.*

$$\mathrm{E}[\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) | A_n] \leq \frac{1}{n} \mathrm{E}\Big[\log \frac{p^*(X^n)}{p_{\mathrm{tp}}^{(n)}(X^n)} | A_n\Big] - \frac{\alpha}{n} \log P_n, \tag{3.1}$$

$$\Pr\Big\{\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) - \frac{1}{n} \log \frac{p^*(X^n)}{p_{\mathrm{tp}}^{(n)}(X^n)} > b | A_n\Big\} \leq \frac{1}{P_n} e^{-n\alpha^{-1}b}. \tag{3.2}$$

Note that this Theorem includes Theorems 2.1 and 2.2 as the case of $A_n = \mathcal{X}^n$. When there exists an upper bound of regret $\overline{\mathrm{REG}}_{A_n}(n)$ such that for all $x^n \in A_n$,

$$\mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}, x^n) \leq \overline{\mathrm{REG}}_{A_n}(n),$$

Theorem 3.1 gives us the following guarantees for the MDL estimator defined by $(\ddot{\mathcal{P}}_n, \alpha L_n)$,

$$\mathrm{E}[\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) | A_n] \leq \frac{1}{n} \overline{\mathrm{REG}}_{A_n}(n) - \frac{\alpha}{n} \log P_n$$

and

$$\Pr\Big\{\bar{d}_\lambda(p^* \| \ddot{p}_{X^n}) - \frac{1}{n} \overline{\mathrm{REG}}_{A_n}(n) > b | A_n\Big\} \leq \frac{1}{P_n} e^{-n\alpha^{-1}b}.$$

3.2 Contaminated Gaussian location families

In this section, we discuss some properties of our contaminated Gaussian location families. We show that our model can satisfies conditions we need to obtain a regret bound, when given

data string belongs to a certain typical set. Then, we can obtain a conditional regret bound which leads to a conditional risk and loss bound by Theorem 3.1.

We consider only the case q_θ is Gaussian location family with unit variance I as follows.

$$q_\theta(x) = \mathcal{N}(x|\theta, I) = \frac{1}{(2\pi)^{K/2}} \exp\Big(-\frac{1}{2}|x-\theta|^2\Big).$$

Further, we assume the contamination is also Gaussian which has the same mean of q_θ and fixed variance $s^2 I$, where $s > 1$ is a known constant.

$$g_\theta(x) = \mathcal{N}(x|\theta, s^2 I) = \frac{1}{(2\pi)^{K/2} s} \exp\Big(-\frac{1}{2s^2}|x-\theta|^2\Big).$$

Our setting on g_θ is easier one than traditional setting in which g_θ is not assumed to have a known expression [19]. However, our model $p_\theta = (1-\nu_c)q_\theta + \nu_c g_\theta$ still has a difficulty for MDL estimation. This kind of model which the contamination has the same location parameter also has been discussed in [21], [20].

This model p_θ is determined by ν_c and s. The following constant T_s defined by s determines the points where $q_\theta(x) = g_\theta(x)$ holds.

$$T_s = \sqrt{2\frac{s^2}{s^2-1}\log s^K},$$

The empirical Fisher information of contamination model $\mathcal{M}$ is

$$\begin{aligned}\hat{J}(\theta; x) = \Big((1-\nu_c)^2\Big(\frac{q_\theta(x)}{p_\theta(x)}\Big)^2 + \frac{\nu_c^2}{s^2}\Big(\frac{g_\theta(x)}{p_\theta(x)}\Big)^2 \\ + \frac{s^2+1}{s^2}\nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)}\Big)I \\ - \Big(\frac{s^2-1}{s^2}\Big)^2\nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)}\hat{\Sigma},\end{aligned} \tag{3.3}$$

where

$$\hat{\Sigma}_{ij} = \hat{\Sigma}_{ij}(\theta; x) = (x_i - \theta_i)(x_j - \theta_j).$$

To see the property of $\hat{J}$, we provide Figure 3.1, a graph of $J(\theta = 0; x)$ for the case of $K = 1$, $s = 5$ and $\nu_c = 10^{-2}$. The vertical lines in Fig.3.1 shows $|x| = T_s$. If and only if $|x-\theta| = T_s$,

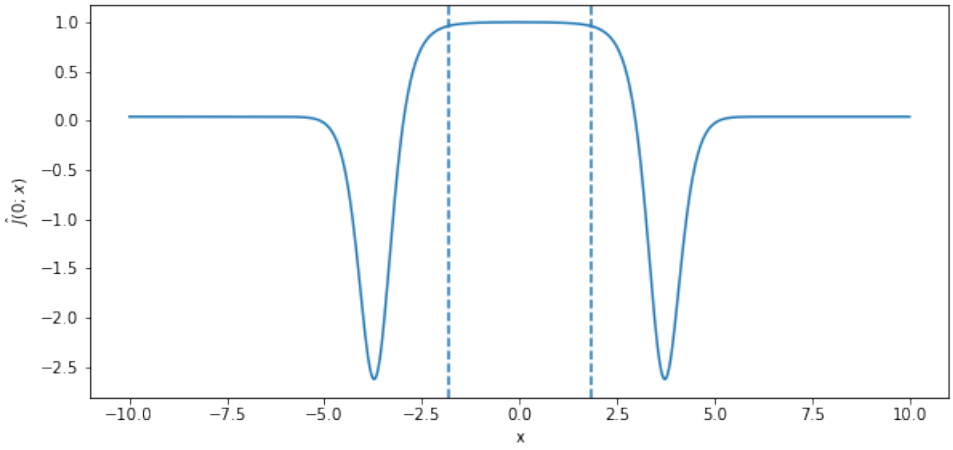

Figure 3.1: A graph of $\hat{J}(0; x)$ when $s = 5$, $\nu_c = 10^{-2}$.

$q_\theta(x) = g_\theta(x)$ holds. Note that $\hat{J}(\theta; x)$ can be negative and then $\hat{J}(\theta; x^n)$ can be close to 0. This causes that the properties given by Lemma 13, which we describe later, breaks. However, $\hat{J}(\theta; x)$ is positive for x within the vertical lines. This implies that if the data x^n includes many points x which are close to the true mean θ^*, $\hat{J}(\theta; x^n)$ can have the positive lower bound for θ close to θ^*. We can show similar properties required by the previous Assumptions for such x^n.

3.3 A typical set of data strings with good properties

In this section, we define A_n a subset of $\mathcal{X}^n$ which characterized by the constant T_s. Then we show that each $x^n \in A_n$ have required properties to show the good regret bound and the probability $P_n = \Pr\{X^n \in A_n\}$ is exponentially high in n. Using this A_n, Theorem 3.1 provides us with the performance guarantee of the MDL estimator.

First, we introduce random variables for $i = 1, 2, \cdots, n$, for given $r \in (0, 1)$,

$$Y_i = \mathbb{1}(|X_i - \theta^*| \leq rT_s),$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. Further, define

$$n_1 = \sum_{i=1}^{n} Y_i.$$

The random variable n_1 denotes the number of x_t in x^n which satisfies $|x_t - \theta^*| \leq rT_s$.

Let $\underline{f} \in (0, 1)$. Define a sequence $\{A_n\}$ by

$$A_n = \left\{ x^n \in \mathcal{X}^n | \frac{n_1}{n} \geq \underline{f} \right\} \tag{3.4}$$

and

$$P_n = \Pr\{X^n \in A_n\} = \int_{A_n} p^*(x^n) dx^n.$$

To show properties of A_n and P_n, we put some conditions on the model. We assume that the parameters of the model ν_c and s satisfy the following conditions.

$$4(1 - \nu_c) \geq \frac{\nu_c}{s} e \tag{3.5}$$

and

$$(1 - \nu_c)(1 + \beta_c)\Phi(T_s) - \beta_c > 0, \tag{3.6}$$

where $\Phi(\cdot)$ is defined by

$$\Phi(d) = \int_{|x| \leq d} \mathcal{N}(x|0, I) dx$$

for all $d \in (0, \infty)$. Note that these conditions hold for sufficiently large s for any small ν_c. For $r \in (0, 1)$, $\Phi(rT_s)$ is a continuous and increasing function of r.and $\Phi(0) = 0$. Therefore under the condition (3.6), there exists $\underline{r} \in (0, 1)$ such that

$$(1 - \nu_c)(1 + \beta_c)\Phi(\underline{r}T_s) - \beta_c = 0 \tag{3.7}$$

and for all $r \in (\underline{r}, 1)$,

$$(1 - \nu_c)(1 + \beta_c)\Phi(rT_s) - \beta_c > 0 \tag{3.8}$$

holds. We also put a condition on the true parameter θ^*. Define

$$\Theta_r = \Big\{\theta \in \mathbb{R}^K \| |\theta| < \frac{1-r}{2} T_s \Big\}.$$

Hereafter, we assume that $\theta^* \in \Theta_r$ so that for all $\theta \in \Theta_r$, $|x - \theta^*| \leq rT_s$ implies $|x - \theta| \leq T_s$.

The following Lemma implies that we can select $\underline{f}$ which defines $\{A_n\}$ with properties we need. In concrete, $\hat{J}(\theta; x^n)$ has properties we need for $x^n \in A_n$ and that this subset of data A_n is a kind of typical set i.e. P_n, which is the probability of the event that $x^n \in A_n$, is exponentially high in n.

Lemma 12. *Under the conditions* (3.5) *and* (3.6)*, for all* $r \in (\underline{r}, 1)$*, where* $\underline{r}$ *is defined by* (3.7)*, and for all* ζ' *such that*

$$0 < \zeta' < B_{s,\nu_c} \min\{1, (1-\nu_c)(1+\beta_c)\Phi(rT_s) - \beta_c\},$$

where

$$B_{s,\nu_c} = (1-\nu_c)^2 - 2\frac{s^2-1}{s^2}\nu_c(1-\nu_c)\log\frac{s^K}{\nu_c}$$

and

$$\beta_c = \frac{1}{2B_{s,\nu_c}}\frac{s^2-1}{s^2}\log\Big(4s^K\frac{1-\nu_c}{\nu_c}\Big),$$

we can select $\underline{f} \in (0,1)$ *so that for all* $x^n \in A_n$*, for all* $\theta \in \Theta_r$ *and for all unit vector* $z \in \mathbb{R}^K$*,*

$$z^T \hat{J}(\theta; x^n) z \geq \zeta' \tag{3.9}$$

holds, further

$$1 - P_n \leq e^{-\frac{1-\nu_c}{2}\gamma_c^2\Phi(rT_s)n}$$

holds, where $\gamma_c \in (0,1)$ *is defined by*

$$\gamma_c = \frac{\beta_c + \zeta'/B_{s,\nu_c}}{(1-\nu_c)(1+\beta_c)\Phi(rT_s)}.$$

Under Assumption 1, this Lemma leads to the following Lemma.

Lemma 13. *There exists constants $\kappa_1, \kappa_2 > 0$ such that for all $\theta_1, \theta_2 \in \Theta_r$, for all unit vector z and for all $x^n \in A_n$, the followings hold.*

$$\frac{z^T \hat{J}(\theta_1; x^n) z}{z^T \hat{J}(\hat{\theta}; x^n) z} \leq 1 + \kappa_1 |\theta_1 - \hat{\theta}| \tag{3.10}$$

and

$$\frac{z^T J(\theta_1) z}{z^T J(\theta_2) z} \leq 1 + \kappa_2 |\theta_1 - \theta_2|.$$

The following Lemma implies that Assumption 2 holds.

Lemma 14. *Under conditions* (3.5) *and* (3.6)*, the followings hold. For all $r \in (\underline{r}, 1)$ and for all $\theta \in \Theta_r$,*

$$\lambda_{\min}(J(\theta)) \geq \zeta,$$

where

$$\zeta = \left\{(1 - \nu_c)(1 + \beta_c)\Phi(rT_s) - \beta_c\right\} B_{s,\nu_c} > 0$$

and there exists $\bar{\lambda} > 0$ which is determined by s and ν_c such that

$$\lambda_{\max}(J(\theta)) \leq \bar{\lambda}.$$

Further, we can show the following Lemma, which implies Assumption 8 holds.

Lemma 15. *Under Assumption 2, there exists $\epsilon > 0$ such that for all unit vector $z \in \mathbb{R}^K$, for all x^n and for all $\theta \in \Theta_r$ such that $|\theta - \hat{\theta}| < \epsilon$,*

$$z^T \hat{J}(\theta; x^n) z < z^T \hat{J}(\hat{\theta}; x^n) z + z^T J(\hat{\theta}) z.$$

Lemmas 12–15 imply that for all $x^n \in A_n$, the regret bound in Theorem 2.4 holds for our contaminated models. Theorem 2.4 gives us an upper bound of the regret of a two part code using a fiber bundle of local exponential families.

Lemma 16. *Under conditions* (3.5) *and* (3.6)*, there exists a constant* $\underline{r} \in (0,1)$ *such that for all* $r \in (\underline{r}, 1)$*, we can construct a two part code which satisfies the following. For large n and for all* $x^n \in A_n$ *such that* $\hat{\theta} \in \Theta_r^\circ$*,*

$$\frac{1}{\alpha}\mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}_r, x^n) \le \frac{K}{2}\log\frac{n}{2\pi} + \log\int_{\Theta_r} |J(\theta)|^{1/2} d\theta + f_{\mathrm{ne}}(n), \tag{3.11}$$

Let $\overline{\mathrm{REG}}(n)$ be the right side of (3.11), By this Lemma and Theorem 3.1, we have the following Theorems.

Theorem 3.2. *Under the same setting as Lemma 16, for all* $\lambda \in (0, 1-\alpha^{-1}]$*, there exist a constant* $r \in (0,1)$ *such that we can construct a two part code which satisfies for large n under the assumption* $\theta^* \in \Theta_r$*,*

$$\mathrm{E}_{X^n \sim p_{\theta^*}}[\bar{d}_\lambda(q_{\theta^*} \| \ddot{p}_{X^n}) | X^n \in A_n] \le \frac{1}{n}\overline{\mathrm{REG}}(n) - \frac{\alpha}{n}\log P_n$$

Theorem 3.3. *Under the same setting as Lemma 16, for all* $\lambda \in (0, 1-\alpha^{-1}]$ *and* $b > 0$*, there exist a constant* $r \in (0,1)$ *and a two part code which satisfies for large n under the assumption* $\theta^* \in \Theta_r$*,*

$$\Pr\Big\{\bar{d}_\lambda(q_{\theta^*} \| \ddot{p}_{X^n}) > \frac{1}{n}\overline{\mathrm{REG}}(n) + b\Big\} \le e^{-n\alpha^{-1}b} + (1 - P_n).$$

From Lemma 12, the probability P_n converges to 1 exponentially.

3.4 Proofs

Proof for Theorem 3.1. We start from (2.61),

$$\sum_{\bar{p} \in \ddot{\mathcal{P}}} \mathrm{E}\exp\Big(\frac{n\bar{d}_\lambda(p^* \| \bar{p})}{\alpha} - \frac{1}{\alpha}\log\frac{p^*(X^n)}{\bar{p}(X^n)} - L_n(\bar{p})\Big) \le 1.$$

By exchanging the summation and the expectation, we have

$$\mathrm{E}\sum_{\bar{p}\in\ddot{\mathcal{P}}}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\bar{p})}{\alpha}-\frac{1}{\alpha}\log\frac{p^*(X^n)}{\bar{p}(X^n)}-L_n(\bar{p})\Big)\le 1.$$

Since $\exp(\cdot)$ is a positive function, this implies

$$\mathrm{E}\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\ddot{p})}{\alpha}-\frac{1}{\alpha}\log\frac{p^*(X^n)}{\ddot{p}(X^n)}-L_n(\ddot{p})\Big)\le 1. \tag{3.12}$$

In general, a conditional expectation of a non-negative random variable satisfies

$$\mathrm{E}[\cdot|A_n]\le\frac{1}{P_n}\mathrm{E}[\cdot].$$

Then, from (3.12), we have

$$\mathrm{E}\Big[\exp\Big(\frac{n\bar{d}_\lambda(p^*\|\ddot{p})}{\alpha}-\frac{1}{\alpha}\log\frac{p^*(X^n)}{\ddot{p}(X^n)}-L_n(\ddot{p})\Big)|A_n\Big]\le\frac{1}{P_n}.$$

By taking logarithms of both side, using Jensen's inequality, and multiplying both side by α/n, we have (3.1).

By using Markov's inequality, we have

$$\Pr\Big\{\bar{d}_\lambda(p^*\|\ddot{p})-\frac{1}{n}\log\frac{p^*(X^n)}{p^{(n)}_{\mathrm{tp}}(X^n)}>\exp(nb\alpha^{-1})\mid A_n\Big\}\le\frac{1}{P_n}e^{-nb\alpha^{-1}}.$$

This is (3.2). □

Proof for Lemma 12. For a real symmetric matrix $A\in\mathbb{R}^{K\times K}$, let $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ denote the maximum and minimum eigenvalue of A respectively. To show (3.9), we evaluate a lower bound of $\lambda_{\min}(\hat{J}(\theta;x^n))$. Since $\hat{\Sigma}$ is semi-positive definite,

$$\lambda_{\max}(\hat{\Sigma})\le\mathrm{Tr}(\hat{\Sigma})=|x-\theta|^2$$

holds. Then, from (3.3), we have for all θ and for all $x\in\mathcal{X}$,

$$\begin{aligned}\lambda_{\min}(\hat{J}(\theta;x))\ge&(1-\nu_c)^2\Big(\frac{q_\theta(x)}{p_\theta(x)}\Big)^2+\frac{\nu_c^2}{s^2}\Big(\frac{g_\theta(x)}{p_\theta(x)}\Big)^2+\frac{s^2+1}{s^2}\nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)}\\&-\Big(\frac{s^2-1}{s^2}\Big)^2\nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)}|x-\theta|^2.\end{aligned} \tag{3.13}$$

To evaluate this lower bound, define

$$v = 1 - \nu_c + \nu_c \frac{g_\theta(x)}{q_\theta(x)}.$$

This variable v takes values over $(1 - \nu_c, \infty)$. From definition, $|x - \theta|^2$ can be written as a function of v as follows.

$$|x - \theta|^2 = 2\frac{s^2}{s^2 - 1} \log\Big(\frac{s^K}{\nu_c} v - s^K \frac{1 - \nu_c}{\nu_c}\Big) \tag{3.14}$$

Further, define

$$f(v) = \Big(\frac{1}{\nu_c v} - \frac{1 - \nu_c}{\nu_c v^2}\Big) \log\Big(\frac{s^K}{\nu_c} v\Big).$$

We show the following Lemma.

Lemma 17. *When ν_c and s satisfy the condition* (3.5)*, the following holds. For all θ and for all $x \in \mathcal{X}$,*

$$\frac{q_\theta(x) g_\theta(x)}{p_\theta^2(x)} |x - \theta|^2 < \frac{1}{2\nu_c(1 - \nu_c)} \frac{s^2}{s^2 - 1} \log\Big(4 s^K \frac{1 - \nu_c}{\nu_c}\Big). \tag{3.15}$$

Proof. First, the followings hold.

$$\begin{aligned} \frac{q_\theta(x)}{p_\theta(x)} &= \frac{1}{1 - \nu_c + \nu_c g_\theta(x)/q_\theta(x)} \\ &= \frac{1}{v} \end{aligned}$$

and

$$\begin{aligned} \frac{g_\theta(x)}{p_\theta(x)} &= \frac{1}{(1 - \nu_c) q_\theta(x)/g_\theta(x) + \nu_c} \\ &= \frac{g_\theta(x)/q_\theta(x)}{1 - \nu_c + \nu_c g_\theta(x)/q_\theta(x)} \\ &= \frac{1}{\nu_c v}\Big(v - (1 - \nu_c)\Big). \end{aligned}$$

Then, we have

$$\frac{q_\theta(x)g_\theta(x)}{p_\theta^2(x)} = \frac{1}{\nu_c v} - \frac{1-\nu_c}{\nu_c v^2}.$$

Therefore, from (3.14),

$$\begin{aligned}\frac{q_\theta(x)g_\theta(x)}{p_\theta^2(x)}|x-\theta|^2 &< 2\frac{s^2}{s^2-1}\Big(\frac{1}{\nu_c v} - \frac{1-\nu_c}{\nu_c v^2}\Big)\log\Big(\frac{s^K}{\nu_c}v\Big) \\ &= 2\frac{s^2}{s^2-1}f(v). \end{aligned} \tag{3.16}$$

Then, we show the following.

$$f(v) \le \frac{1}{4\nu_c(1-\nu_c)}\log\Big(4s^K\frac{1-\nu_c}{\nu_c}\Big). \tag{3.17}$$

Let $T > 1-\nu_c$. We evaluate upper bounds of $f(v)$ for $v \in (1-\nu_c, T]$ and $v > T$ respectively. First, we evaluate an upper bound for $v \in (1-\nu_c, T]$. Since in this case,

$$\frac{1}{\nu_c v} - \frac{1-\nu_c}{\nu_c v^2} > 0$$

holds, we have

$$f(v) \le \Big(\frac{1}{\nu_c v} - \frac{1-\nu_c}{\nu_c v^2}\Big)\log\Big(\frac{s^K}{\nu_c}T\Big).$$

Further since for all $v > 0$,

$$\frac{1}{\nu_c v} - \frac{1-\nu_c}{\nu_c v^2} \le \frac{1}{4\nu_c(1-\nu_c)}$$

holds, we have for all $v \in (1-\nu_c, T]$,

$$f(v) \le \frac{1}{4\nu_c(1-\nu_c)}\log\Big(\frac{s^K}{\nu_c}T\Big).$$

Next, we evaluate an upper bound for $v > T$. Since $(1-\nu_c)/\nu_c v^2 > 0$, for all $v > 0$,

$$f(v) < \frac{1}{\nu_c v}\log\Big(\frac{s^K}{\nu_c}v\Big)$$

holds. The right side of this is a decreasing function of v for $v > (\nu_c/s)e$. Then when we select $T \geq (\nu_c/s)e$, the following holds. For all $v > T$,

$$f(v) < \frac{1}{\nu_c T} \log\Big(\frac{s^K}{\nu_c} T\Big).$$

Therefore, for all $v > 1 - \nu_c$,

$$f(v) \leq \max\Big\{\frac{1}{4\nu_c(1-\nu_c)} \log\Big(\frac{s^K}{\nu_c} T\Big), \frac{1}{\nu_c T} \log\Big(\frac{s^K}{\nu_c} T\Big)\Big\}$$

holds. By letting $T = 4(1 - \nu_c)$ to minimize the right side, we have (3.17). From (3.16) and (3.17), we have (3.15). □

When $|x - \theta| \leq T_s$, $v \leq 1$ holds. Since $f(v)$ is an increasing function for $v \leq 2(1 - \nu_c)$, for all $v \leq 1$,

$$f(v) \leq f(1) = \log\Big(\frac{s^K}{\nu_c}\Big)$$

holds. Therefore, by ignoring the second and third terms in (3.13), since $q_\theta(x) \geq g_\theta(x)$ for $|x - \theta| \leq T_S$, for all $x \in \mathcal{X}$ and $\theta \in \Theta$ such that $|x - \theta| \leq T_s$,

$$\lambda_{\min}(\hat{J}(\theta; x)) \geq B_{s,\nu_c} \tag{3.18}$$

holds. Further when $|x - \theta| > T_s$, from Lemma 17 and by ignoring the terms except the last one in (3.13), we have

$$\lambda_{\min}(\hat{J}(\theta; x)) \geq -\beta_c B_{s,\nu_c}. \tag{3.19}$$

When $\theta^* \in \Theta_r$, for arbitrary $\theta \in \Theta_r$

$$|\theta - \theta^*| \leq (1 - r)T_s$$

holds. Then, for all x such that $|x - \theta^*| \leq rT_s$,

$$|x - \theta| \leq T_s$$

also holds. Therefore, for all $x^n \in \mathcal{X}^n$ and for all $\theta \in \Theta_r$, since at least n_1 data in x^n satisfies $|x - \theta| \le T_s$,

$$\lambda_{\min}(\hat{J}(\theta; x^n)) \ge \frac{n_1}{n} B_{s,\nu_c} - \Big(1 - \frac{n_1}{n}\Big)\beta_c B_{s,\nu_c}$$

holds. Let $\zeta' \in (0, B_{s,\nu_c})$. Define $\underline{f} \in (0, 1)$ as

$$\underline{f} = 1 - \frac{B_{s,\nu_c} - \zeta'}{(1 + \beta_c)B_{s,\nu_c}}.$$

Then, define A_n by (3.4). In this case, we have for all $x^n \in A_n$,

$$\lambda_{\min}(\hat{J}(\theta; x^n)) \ge \zeta'.$$

This implies (3.9).

Finally, we evaluate the probability $P_n = \Pr\{X^n \in A_n\}$. Since

$$\begin{aligned}\Pr\{|X - \theta^*| \le rT_s\} &= (1 - \nu_c) \int_{|x-\theta^*| \le rT_s} q_{\theta^*}(x)dx \quad + \nu_c \int_{|x-\theta^*| \le rT_s} g_{\theta^*}(x)dx \\ &\ge (1 - \nu_c)\Phi(rT_s),\end{aligned}$$

we have

$$\mathrm{E}n_1 \ge n(1 - \nu_c)\Phi(rT_s).$$

Define

$$\gamma_c = 1 - \frac{\beta_c + \zeta'/B_{s,\nu_c}}{(1 - \nu_c)(1 + \beta_c)\Phi(rT_s)}.$$

When ν_c and s satisfy

$$\Phi(rT_s) > \frac{1}{1 - \nu_c}\frac{\beta_c}{1 + \beta_c}$$

and we select ζ' so that

$$0 < \zeta' < B_{s,\nu_c}\{(1 - \nu_c)(1 + \beta_c)\Phi(rT_s) - \beta_c\},$$

$\gamma_c \in (0, 1)$ holds. Therefore, from Chernoff bound, we have

$$1 - P_n = \Pr\{X^n \notin A_n\} \leq e^{-\frac{1-\gamma_c}{2}\gamma_c^2 \Phi(rT_s)n}.$$

□

Proof for Lemma 13. By Taylor expansion, for all $\theta_1, \theta_2 \in \Theta_r$, for all x^n and for all unit vector $z \in \mathbb{R}^K$, there exists θ' between θ_1 and θ_2 such that

$$z^T \hat{J}(\theta_1; x^n) z = z^T \hat{J}(\theta_2) z + \sum_{h,i,j} \frac{\partial}{\partial \theta_h} \hat{J}_{ij}(\theta; x^n)\Big|_{\theta=\theta'} z_i z_j (\theta_1 - \theta_2)_h,$$

where $(\theta_1 - \theta_2)_h$ denotes h-th elements of $\theta_1 - \theta_2$. From Cauchy-Schwarz inequality, for all $v \in \mathbb{R}^K$,

$$\sum_i v_i \leq \sqrt{K}|v|$$

holds. Further from Assumption 1, there exists a constant $D > 0$ such that for all $h, i, j \in \{1, 2, \cdots, K\}$, for all $\theta \in \Theta$ and for all x^n,

$$\left|\frac{\partial}{\partial \theta_h} \hat{J}_{ij}(\theta; x^n)\right| \leq D.$$

Therefore, we have

$$z^T \hat{J}(\theta_1; x^n) z \leq z^T \hat{J}(\theta_2; x^n) z + K\sqrt{K} D |\theta_1 - \theta_2|.$$

From Lemma 12, since for all $x^n \in A_n$, $z^T \hat{J}(\theta_1; x^n) z \geq \zeta' > 0$ holds, we have

$$\frac{z^T \hat{J}(\theta_1; x^n) z}{z^T \hat{J}(\theta_2; x^n) z} \leq 1 + \frac{K\sqrt{K}}{\zeta'} D |\theta_1 - \theta_2|.$$

By letting $\theta_2 = \hat{\theta}$ and $\kappa_1 = K\sqrt{K}D/\zeta'$, we have (3.10).

Assumption 1 also implies that

$$\left|\frac{\partial}{\partial \theta_h} J_{ij}(\theta)\right| \leq D.$$

Therefore under Assumption 2, we have

$$\frac{z^T J(\theta_1) z}{z^T J(\theta_2) z} \leq 1 + \frac{K\sqrt{K}}{\zeta} D|\theta_1 - \theta_2|$$

in the same manner. □

Proof for Lemma 14. Fisher information is defined by

$$\begin{aligned} J(\theta) &= \mathrm{E}_{p_\theta} \hat{J}(\theta; X) \\ &= \int_{|x-\theta| \leq rT_s} \hat{J}(\theta; x) p_\theta(x) dx + \int_{|x-\theta| > rT_s} \hat{J}(\theta; x) p_\theta(x) dx. \end{aligned}$$

Then,

$$\begin{aligned} \lambda_{\min}(J(\theta)) \geq \Pr\{|X-\theta| \leq rT_s\} &\inf_{|x-\theta| \leq rT_s} \lambda_{\min}(\hat{J}(\theta; x)) \\ &+ \Pr\{|X-\theta| > rT_s\} \inf_{|x-\theta| > rT_s} \lambda_{\min}(\hat{J}(\theta; x)) \end{aligned}$$

holds. From (3.18) and (3.19), we have

$$\begin{aligned} \lambda_{\min}(J(\theta)) &\geq \Pr\{|X-\theta| \leq rT_s\} B_{s,\nu_c} - \Pr\{|X-\theta| > rT_s\} \beta_c B_{s,\nu_c} \\ &\geq \Big\{(1-\nu_c)\Phi(rT_s)(1+\beta_c) - \beta_c\Big\} B_{s,\nu_c}. \end{aligned}$$

For all $r \in (\underline{r}, 1)$, from (3.8), this right side is positive.

To show the upper bound of $\lambda_{\max}(J(\theta))$, we use the fact that

$$\lambda_{\max}(J(\theta)) \leq \|J(\theta)\|_s \leq K \|J(\theta)\|_M.$$

It is sufficient to show an upper bound of $\|J(\theta)\|_M$. Since $J(\theta) = \mathrm{E}_{p_\theta} \hat{J}(\theta; X)$, $\|J(\theta)\|_M \leq \max_{\theta,x} \|\hat{J}(\theta; x)\|_M$ holds. Therefore, we show an upper bound of $\|\hat{J}(\theta; x)\|_M$. In the same manner as (3.13),

$$\begin{aligned} \lambda_{\max}(\hat{J}(\theta; x)) \leq &(1-\nu_c)^2 \Big(\frac{q_\theta(x)}{p_\theta(x)}\Big)^2 + \frac{\nu_c^2}{s^2}\Big(\frac{g_\theta(x)}{p_\theta(x)}\Big)^2 + \frac{s^2+1}{s^2}\nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)} \\ &+ \Big(\frac{s^2-1}{s^2}\Big)^2 \nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)}|x-\theta|^2 \end{aligned}$$

holds. Further since $q_\theta(x)/p_\theta(x) \le 1/(1-\nu_c)$ and $g_\theta(x)/p_\theta(x) \le 1/\nu_c$ hold, we have

$$\lambda_{\max}(\hat{J}(\theta;x)) \le 2\frac{s^2+1}{s^2} + \Big(\frac{s^2-1}{s^2}\Big)^2 \nu_c(1-\nu_c)\frac{q_\theta(x)}{p_\theta(x)}\frac{g_\theta(x)}{p_\theta(x)}|x-\theta|^2.$$

Therefore, from Lemma 17, we have

$$\lambda_{\max}(\hat{J}(\theta;x)) \le 2\frac{s^2+1}{s^2} + \frac{1}{2}\frac{s^2-1}{s^2}\log\Big(4s^K\frac{1-\nu_c}{\nu_c}\Big). \tag{3.20}$$

From (3.18), (3.19) and (3.20), we have

$$\|\hat{J}(\theta;x)\|_M \le \hat{\lambda}',$$

where

$$\hat{\lambda}' = \max\Big\{B_{s,\nu_c}, \beta_c B_{s,\nu_c}, 2\frac{s^2+1}{s^2} + \frac{1}{2}\frac{s^2-1}{s^2}\log\Big(4s^K\frac{1-\nu_c}{\nu_c}\Big)\Big\}.$$

Finally, we have

$$\lambda_{\max}(J(\theta)) \le \bar{\lambda},$$

where

$$\bar{\lambda} = K\hat{\lambda}'.$$

□

Proof for Lemma 15. From (3.3), the partial derivatives with θ of $\hat{J}(\theta;x)$ are also polynomial of $q_\theta(x)/p_\theta(x)$, $g_\theta(x)/p_\theta(x)$ and $(q_\theta(x)g_\theta(x)/p_\theta(x)^2)(x-\theta)^m$, where $m = 2$ or 3. Then, in the same manner as Lemma 17, we can show that there exists a certain constant $D_{\hat{J}} > 0$ such that for all unit vector $z, v \in \mathbb{R}^K$, for all $\theta \in \Theta_r$ and for all $x^n \in \mathcal{X}^n$,

$$|\nabla_v z^T \hat{J}(\theta) z| \le D_{\hat{J}}.$$

Therefore, by Taylor expansion, we have

$$z^T \hat{J}(\theta; x^n) z \le z^T \hat{J}(\hat{\theta}; x^n) z + D_{\hat{f}} |\theta - \hat{\theta}|.$$

Since $z^T J(\hat{\theta}) z > \zeta$, when

$$|\theta - \hat{\theta}| < \frac{\zeta}{D_{\hat{f}}},$$

we have

$$z^T \hat{J}(\theta; x^n) z < z^T \hat{J}(\hat{\theta}; x^n) z + z^T J(\hat{\theta}) z.$$

□

3.5 Concluding remarks

We showed that we can construct a two part code for contaminated Gaussian location families whose parameters satisfy the conditions (3.5) and (3.6), such that for appropriately selected $r \in (0, 1)$, the following regret bound holds for all $x^n \in A_n$ such that $\hat{\theta} \in \Theta_r^\circ$.

$$\frac{1}{\alpha} \mathrm{REG}(p_{\mathrm{tp}}^{(n)}; \mathcal{M}_r, x^n) \le \frac{K}{2} \log \frac{n}{2\pi} + \log \int_{\Theta_r} |J(\theta)|^{1/2} d\theta + f_{\mathrm{ne}}(n).$$

This set $A_n \subset \mathcal{X}^n$ defined by (3.4) is a kind of typical set. That is the probability of the event $X^n \in A_n$ converges to 1 exponentially (Lemma 12). Therefore, the regret bound holds with high probability for large n.

By using Theorem 3.1, the regret bound leads to the following conditional risk and loss bounds of the MDL estimator induced by the two part code

$$\mathrm{E}_{X^n \sim p_{\theta^*}} [\bar{d}_\lambda(q_{\theta^*} \| \ddot{p}_{X^n}) | X^n \in A_n] \le \frac{1}{n} \overline{\mathrm{REG}}(n) - \frac{\alpha}{n} \log P_n$$

and

$$\Pr\left\{ \bar{d}_\lambda(q_{\theta^*} \| \ddot{p}_{X^n}) > \frac{1}{n} \overline{\mathrm{REG}}(n) + b \right\} \le e^{-n\alpha^{-1} b} + (1 - P_n).$$

Since A_n is a typical set, these bounds are sufficiently tight for large n.

Chapter 4

On Jeffreys factor of Gaussian Distributions

The topic of this chapter is quite different from the previous chapters. In this chapter, we consider the integral which appears in the evaluation of minimax regret of parametric models. We focus on Gaussian distributions. There is an important example of non-exponential family which is closely related to Gaussian distributions, called Gaussian mixture models (GMM). We expect that the discussion in this chapter may be helpful to consider two part codes for GMM.

4.1 Introduction

Gaussian Mixture Model (GMM) is an example of non-exponential families, which is important in practice and more complicated than mixture families. A density of GMM is represented as a mixture of Gaussian distributions, which have identical parameters respectively. The difference between GMM and mixture families, whose components are Gaussian, is that each components of mixture has parameters respectively. Then, when we fix the parameters of each components, we can recognize the model as a mixture family.

In this chapter, first, we discuss the determinant of Fisher information of multi-variate Gaussian distributions parametrized with mean μ and covariance Σ, which we need to quantizing the parameter space of multi-variate Gaussian distributions. Then, we discuss the integral

of the square root of the determinant. This integral appears in the evaluation of minimax regret (2.2) and also in the regret bound of two part codes introduced in the previous chapters. In conclusion, we can see that the restriction of the range of eigenvalues of covariance Σ is essentially required.

4.2 Parametrization for covariance of Gaussian distributions

Let $K > 0$ be an integer. The probability density function of Gaussian distribution with mean $\mu \in \mathbb{R}^K$ and covariance $\Sigma \in \mathbb{R}^{K\times K}$, which is semi-positive and symmetric, is written as

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{K/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}.$$

These parameters μ and Σ have a degree of freedom K and $K(K-1)/2$ respectively.

The covariance Σ is restricted to positive definite and symmetric matrices. This restriction makes it difficult to consider Fisher information with respect to Σ. We introduce an alternative parametrization to Σ. We use the fact that any positive definite and symmetric matrix Σ can be diagonalized as

$$\Sigma = V^T\Lambda V, \tag{4.1}$$

where Λ is a positive valued diagonal matrix whose elements are an eigenvalue of Σ, and V is an orthogonal matrix. When a positive valued diagonal matrix Λ and an orthogonal matrix V are given, they determine a certain positive definite and symmetric matrix Σ by (4.1). Now, Λ can be parametrized directly by m eigen values $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_m$. Further, V can be represented as a product of $K(K-1)/2$ Givens rotation matrices defined as follows. For each

$k, l \in \{1, 2, \cdots, K\}$ and $\beta \in (0, \pi)$,

$$G_{ij}(k, l, \beta) = \begin{cases} \cos\beta & \text{if } i = j = k \text{ or } i = j = l, \\ \sin\beta & \text{if } i = k,\ j = l, \\ -\sin\beta & \text{if } i = l,\ j = k, \\ 1 & \text{if } i = j,\ i \neq k, l, \\ 0 & \text{otherwise.} \end{cases}.$$

The matrix $G(k, l, \beta)$ is an orthogonal matrix which represents the rotation on (k, l) plane and β means an angle of the rotation.

Define

$$S(K) = \{(k, l) | k, l \in \{1, 2, \cdots, K\}, k < l\}.$$

This can be regarded as the set of all distinct planes spanned by two axes of $\mathbb{R}^K$. The number of such planes is $|S(K)| = K(K-1)/2$. By using appropriate sequences $\{k_t\}$ and $\{l_t\}$, $S(K)$ can be represented as

$$S(K) = \cup_{t=1}^{K(K-1)/2} \{(k_t, l_t)\}.$$

Then, we define for each $t = 1, 2, \cdots, K(K-1)/2$,

$$G_t(\beta) = G(k_t, l_t, \beta).$$

Finally, for $\phi = (\phi_1, \cdots, \phi_{K(K-1)/2}) \in (0, \pi)^{K(K-1)/2}$, we define

$$G(\phi) = \prod_{t=1}^{K(K-1)/2} G_t(\phi_t).$$

This provides a parametrization for orthogonal matrices.

Now, we have a parametrization of covariance matrix Σ by using eigenvalues $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_K)$ and $\phi \in (0, \pi)^{K(K-1)/2}$,

$$\Sigma(\Lambda, \phi) = G(\phi)^T \Lambda G(\phi).$$

Define a composite parameter $\theta = \theta(\mu, \Lambda, \phi)$ as

$$\theta_i = \mu_i \text{ for } i = 1, 2, \cdots, K,$$

$$\theta_{K+i} = \lambda_i \text{ for } i = 1, 2, \cdots, K,$$

$$\theta_{2K+i} = \phi_i \text{ for } i = 1, 2, \cdots, K(K-1)/2.$$

This defines a parametrization of Gaussian distributions,

$$p_\theta(x) = \frac{1}{(2\pi)^{K/2}|\Lambda|^{-1/2}} \exp\Big\{-\frac{1}{2}(x-\mu)^T G(\phi)\Lambda^{-1}G(\phi)^T(x-\mu)\Big\}.$$

We can assume that the eigenvalues $\lambda_1, \cdots \lambda_K$ are in the ascending order,

$$\lambda_1 \le \lambda_2 \le \cdots \le \lambda_K.$$

This parametrization ahs an advantage that we can use eigenvalues as a part of parameter directly. Then by letting all eigenvalues be positive, the positive definiteness of Σ is ensured.

4.3 Fisher information of Gaussian distribution

For our parametrization, the derivative of $-\log p_\theta(x)$ with respect to μ is

$$-\frac{\partial}{\partial\mu}\log p_\theta(x) = -\Sigma^{-1}(x-\mu). \tag{4.2}$$

Therefore, a part of the empirical Fisher information and Fisher information with respect to μ is Σ^{-1}. Since (4.2) has a factor $x-\mu$ whose expectation is 0 and another factor Σ^{-1} does not depend on x, the expectation of the second derivative with respect to μ and other parameters is 0. This means that cross components of Fisher information with respect to μ and others are 0.

To calculate the Fisher information with respect to other parameters, it is easy to use an alternative representation of the Fisher information by using Kullback-Leibler divergence,

$$J_{ij}(\theta) = \frac{\partial^2}{\partial\theta'_i\partial\theta'_j} D_{KL}(p_\theta||p_{\theta'})\Big|_{\theta'=\theta}.$$

It is known that the Kullback-Leibler divergence between Gaussian distributions can be written as

$$D_{KL}(p_\theta||p_{\theta'}) = \frac{1}{2}\log\frac{|\Lambda'|}{|\Lambda|} - \frac{K}{2} + \frac{1}{2}\mathrm{Tr}\{\Sigma'^{-1}(\Sigma + (\mu-\mu')^T(\mu-\mu'))\}.$$

Since

$$\frac{\partial^2}{\partial\lambda'_i\partial\lambda'_j}\log|\Lambda'| = -\frac{1}{\lambda_i^2}I_{ij}$$

and

$$\mathrm{Tr}\Big\{\frac{\partial^2}{\partial\lambda'_i\partial\lambda'_j}\Sigma'^{-1}\Big|_{\theta'=\theta}\Sigma\Big\} = \frac{2}{\lambda_i^2}I_{ij}$$

hold, a part of the Fisher information with respect to eigenvalues is a diagonal matrix

$$J_\Lambda(\theta) = \mathrm{diag}\Big(\frac{1}{2\lambda_1^2},\cdots,\frac{1}{2\lambda_K^2}\Big).$$

Next, we calculate a part of the Fisher information with respect to ϕ. For general ϕ, it is complicated to calculate the Fisher information. However for each ϕ, when we introduce a transformation $Y_\phi = G(\phi)X$, the distribution of Y_ϕ is $\mathcal{N}(G(\phi)\mu,\Lambda)$. Since the Fisher information is invariant for such transformation, it is sufficient to evaluate the Fisher information of Y_ϕ whose covariance is diagonal. This implies that we need only to consider the case of $\phi = 0$.

For $t = 1,\cdots,K$, let E_t be the diagonal matrix whose k_t-th and l_t-th elements are 0 and others are 1. The derivatives of Gives rotation matrices can be denoted as

$$\frac{\partial}{\partial\phi_t} = G_t(\theta_t + \frac{\pi}{2}) - E_t,$$
$$\frac{\partial^2}{\partial\phi_t^2} = -G_t(\theta_t) + E_t.$$

Then,

$$\frac{\partial}{\partial\phi'_i}G(\phi')\Big|_{\phi'=0} = G_t(\frac{\pi}{2}) - E_i,$$
$$\frac{\partial^2}{\partial\phi'_i\partial\phi'_j}G(\phi')\Big|_{\phi'=0} = (G_i(\frac{\pi}{2}) - E_i)(G_j(\frac{\pi}{2}) - E_j)$$

holds. Let $E'_t = G_t(\pi/2) - E_t$. For each t, the $k_t l_t$ element of E'_t is -1, the $l_t k_t$ element is 1 and other elements are 0. Then when $\phi = 0$, the second derivatives of Σ^{-1} with respect to ϕ can be denoted as follows.

$$\frac{\partial}{\partial\phi'_i\partial\phi'_j}\Sigma'^{-1}\Big|_{\phi'=0} = E'_i E'^T_j \Lambda^{-1} - E'_i\Lambda^{-1}E'^T_j - E'_j\Lambda^{-1}E'^T_i + \Lambda^{-1}E'_i E'^T_j.$$

By using this, we can calculate the trace

$$\frac{1}{2}\mathrm{Tr}\Big\{\frac{\partial^2}{\partial\phi'_i\partial\phi'_j}\Sigma'^{-1}\Big|_{\phi'=0}\Lambda\Big\} = \begin{cases} \frac{(\lambda_{l_i}-\lambda_{k_i})^2}{\lambda_{k_i}\lambda_{l_i}} & \text{for } i = j, \\ 0 & \text{for } i \neq j \end{cases}. \tag{4.3}$$

Therefore, a part of the Fisher information with respect to ϕ is a diagonal matrix $J_\phi(\theta)$ whose elements are given by (4.3).

4.4 On Jeffreys factor for the parametrization using eigenvalues

Now, we have the Fisher information in the form of block diagonal matrix

$$J(\theta) = \mathrm{diag}(\Sigma^{-1}, J_\Lambda(\theta), J_\phi(\theta)).$$

Then, we have the determinant

$$\begin{aligned} |J(\theta)| &= |\Sigma^{-1}| \times |J_\Lambda(\theta)| \times |J_\phi(\theta)| \\ &= \frac{1}{2^K}\frac{\prod_{t=1}^{K(K-1)/2}(\lambda_{l_t} - \lambda_{k_t})^2}{(\lambda_1\cdots\lambda_K)^{K+2}}. \end{aligned} \tag{4.4}$$

Finally, by using the result, we discuss the normalization factor of Jeffreys prior. By integrating the square root of (4.4), we can obtain the normalization factor of Jeffreys prior. However, the calculation for general $K \geq 2$ is too complicated. Then, we show only the result for $K = 2$. Since (4.4) depends on eigenvalues only, we calculate only the integral with respect to λ_1 and λ_2. To calculate the integral, by using constants $0 < \lambda_{\min} < \lambda_{\max}$, we limit the range of integral to $\lambda_{\min} \leq \lambda_1 \leq \lambda_2 \leq \lambda_{\max}$. We have the double integral

$$\int_{\lambda_{\min}}^{\lambda_{\max}}\Big(\int_{\lambda_{\min}}^{\lambda_2}|J(\theta)|^{1/2}d\lambda_1\Big)d\lambda_2 = \frac{1}{2}\Big(\frac{1}{\lambda_{\max}} + \frac{1}{\lambda_{\min}}\Big)\log\frac{\lambda_{\max}}{\lambda_{\min}} - \Big(\frac{1}{\lambda_{\min}} - \frac{1}{\lambda_{\max}}\Big).$$

When $\lambda_{\max} \to \infty$ or $\lambda_{\min} \to 0$, this integral diverges to ∞. Then, the Jeffreys prior of Gaussian distributions cannot be defined properly without such restriction for eigenvalues of covariance.

Chapter 5

A Practical Study on Machine Learning Based Cyber Security

The increased usage of the Internet and network-connected systems has exposed them to a greater number of cyber threats. These threats are becoming more sophisticated, and the global cybersecurity landscape has seen a steady increase in reported incidents in recent years [24, 25]. These incidents range from conventional attacks such as ransomware, phishing, denial of services (DoS), to novel, hybrid, and emerging threats that have a strong impact on the infrastructure and digital assets managed by organizations.

To protect against these threats, intrusion detection is an indispensable technique for maintaining enterprise security. Network Intrusion Detection Systems (NIDSs) [26, 27] are used to monitor networks or systems and raise alerts when abnormal activities or policy violations are detected. NIDSs use a variety of techniques such as signature-based detection and anomaly-based detection to identify potential threats. When an NIDS detects a threat, it raises an alert to notify the security team, allowing the team to take immediate action to prevent the attack from causing any further harm.

As cyberattacks continue to evolve, they are becoming increasingly diverse and complex, making them harder to detect and defend against. To effectively counter these evolving threats, adaptive NIDSs are highly recommended. Among the different types of NIDSs, machine learn-

ing (ML)-based systems have shown particular promise in detecting novel attacks by analyzing network traffic generated by these attacks and adapting their detection strategies accordingly. One significant advantage of ML-based NIDSs is their ability to learn from data without requiring manual rule design by security analysts. This allows for faster and more accurate detection of novel attacks, enabling organizations to keep up with the rapidly changing threat landscape.

The detection of network intrusions relies on the analysis of communication information transmitted between attacking and target hosts through packets. As the smallest unit of network communication, packets carry crucial control and payload information and are the primary analysis target for intrusion detection. ML-based NIDSs can be categorized into two types based on their analysis target. The first type uses packets directly as the analysis and alert triggering objects, resulting in each alert corresponding to a single packet [28, 29]. The second type aggregates packets into sessions that contain more communication information and uses these sessions as the analysis and alert triggering objects [30, 31].

Packet-level analysis has inherent limitations [32]. Firstly, the sheer number of packets can be overwhelming, resulting in a significant number of false positive alerts that can be challenging to manage. Secondly, not all packets contain essential information for intrusion detection. For example, control packets used to establish TCP connections carry specified information about communication tools, but not particular information about the communication contents itself, making them difficult to classify. To address these limitations, session-level analysis can provide a more comprehensive picture of network communication and help reduce false positive alerts. However, session-level analysis can be challenging to implement, requiring the tracking of packets across multiple network connections and handling complex communication scenarios.

In this paper, we present a novel solution to overcome the drawbacks of packet-level analysis in network intrusion detection. Our proposed approach defines a session as a collection of consecutive packets that participate in communication and leverages the packet-level classification outcomes to identify whether the session is normal or anomalous. By reducing the number of targets for classification, our approach leads to more accurate and reliable predictions based on all packets that contribute to the communication. Furthermore, our approach

does not rely on session-level characteristics, which enables the adoption of various packet-level classification techniques and effortless integration of new features or classifiers.

The paper presents the following contributions:

1. We introduce a novel approach to consolidate packet-level classification outputs to make intrusion detection predictions for communication sessions rather than doing session-level classification directly. Our approach not only effectively reduces the number of alerts but also improves the accuracy of the detection. Further, our approach is independent to how the packet-level prediction is obtained. Therefore, we can use arbitrary packet-level methods with our proposed approach.

2. We develop a prototype system for the proposed approach, which uses statistical features extracted from packet headers.

3. We evaluate the effectiveness of our approach on publicly available benchmark datasets, and report promising results that demonstrate the efficacy of our approach.

Overall, our proposed approach and prototype system offer a practical and effective solution to improve the accuracy and reliability of network intrusion detection, which is increasingly important in the face of evolving and complex cyber threats.

The rest of this paper is structured as follows: Section 5.1 provides a comprehensive survey of intrusion detection techniques that use ML-based methods and our differentiation strategy compared to related works. Section 5.2 describes our approach for network intrusion detection using packet-level features. Section 5.2.1 describes methods for the packet-level classification and extraction of packet-level features. Section 5.2.2 presents the proposed approach for consolidating packet-level classification outputs to predict session-level anomalies. Section 5.3 reports on experimental results from publicly available benchmark datasets. Finally, Section 5.6 concludes the paper.

5.1 Related Work

The use of ML-based NIDS in cybersecurity has grown significantly over the past decade. Researchers have focused on developing more advanced machine learning algorithms such as decision trees, support vector machines, and neural networks for anomaly detection and classification. In recent years, deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have shown significant improvements in the accuracy of IDS [33, 34]. Ensemble learning and transfer learning have emerged as promising approaches to enhance the detection rate and reduce false alarms. Researchers are also collecting large-scale cybersecurity datasets and exploring alternative evaluation routes [35]. Additionally, the focus is on developing more efficient and accurate IDS that can detect evolving and sophisticated attacks in real-time while addressing scalability, cost-effectiveness, and transparency issues. There is a growing emphasis on cloud-based solutions, which are more scalable and cost-effective [36].

5.1.1 Data Type and Public Dataset

ML-based NIDS relies on several data types to detect and classify attacks accurately. Two primary types of data that researchers use are packet-level and session-level data.

- Packet-level data includes information from individual packets within a network flow. Packet-level data can be further divided into header and payload information, where the header includes information such as source and destination IP addresses, port numbers, and protocol types. In contrast, the payload includes the actual content of the packet.
- Session-level data includes data from the sequence of packets between two network nodes over a period of time. It can provide insights into the behavior of network users, applications, and services. It can be used to identify traffic patterns that indicate a botnet or other malicious activity.

The use of either packet-level or session-level data in ML-based NIDS research varies depending on the research objectives and the desired level of granularity. Researchers choose

which type of data to use based on the requirements of the model and the specific objectives of the research. In summary, using different data types in ML-based NIDS research enables researchers to develop more accurate, efficient, and effective models to detect and classify network intrusions more accurately and in real-time.

Next, public datasets of network traffic have been extensively used by researchers to evaluate their models and compare their results with those of other studies. These datasets comprise large-scale network traffic data from different sources, such as honeypots, computer networks, and other online services. Public datasets are essential for developing NIDS because they provide a standardized evaluation framework, facilitating the reproducibility of results across different research groups. In [37], a comprehensive survey of 34 different public datasets was conducted, focusing on 15 different characteristics (data format, data volume, traffic observation environment, etc.). In [34], the authors examined 30 different public datasets in detail, focusing on the types of attacks that each dataset contains. Prominent datasets include the long-used DARPA98 [38,39], KDD99 [40], and NSL-KDD [41], as well as UNSW-NB15 [42], ISCX2012 [43], Kyoto2006+ [44], CTU13 [45], and CSE-CIC-IDS2018 [46].

5.1.2 ML-based NIDS by Data Type

In this section, we present several related studies of ML-based NIDS by the type of input data (packet-level, session-level, and hybrid) used in the analysis. Each related study is summarized in Table 5.1, which lists the data type and features used, datasets used, training models, evaluation results, and any other unique contributions. We use the abbreviations for kinds of ML models in the table. (AE: AutoEncoder, DT: Decision Tree, NB: Naive Bayes, SGD: Stochastic Gradient Descent, PCA: Principal Components Analysis, IF: Isolation Forest, RF: Random Forest, SVM: Support Vector Machine, k-NN: k-Nearest Neighbor, ENN: Elman Neural Network, SNN: Shallow Neural Network, FFNN: FeedForward Neural Network, MLP: MultiLayer Perceptron)

Packet-Level Data Analysis

Here we present related works on ML-based NIDS that analyze packet-level data. Packet-level data can be divided into two categories: either only header information is used, or payload information is also used.

Mirsky *et al.* [28] proposed a framework, Kitsune, that uses an ensemble of autoencoders to differentiate between normal and abnormal patterns. The results showed that Kitsune could detect various attacks with performance comparable to existing offline algorithms, even when running on low-power devices like Raspberry Pi. Hwang *et al.* [47] argued that detecting malicious traffics at the packet level, rather than through flow-based preprocessing, reduces detection time significantly, making it suitable for real-time analysis requirements. Gwon *et al.* [48] used embedding techniques to embed time-series and categorical information, achieved a binary classification accuracy of 99.72%, and significantly improved performance. Shrestha *et al.* [49] evaluated various machine learning models, of which the decision tree algorithm performed best with a maximum accuracy of 99.99% and a minimum false negative of 0%. Verkerken *et al.* [50] demonstrated that all machine learning models they tried could exhibit high classification scores on an individual dataset but could not maintain those scores when applied to a second, unseen but related dataset, prompting the need for further research on techniques to improve the generalizability of the models.

Header information is easy to handle as feature values because its format and value range are uniquely defined. On the other hand, payload information is of variable size and requires ingenuity to treat it as feature values. Conventional methods for handling payload information mainly use simple statistics such as payload length and entropy, losing a lot of information [51,52]. Recently, with the development of natural language processing-based embedding techniques, research on embedding payloads into low-dimensional vectors while preserving the content information (semantics) of the payload has become popular [29, 53].

Session-level Data Analysis

Next, we surveyed related studies of ML-based NIDS that analyze session-level data. There are three cases of analyzing session-level data: when the public dataset is session-level data in the first place [54, 55], as in KDD99 and NSL-KDD; when NetFlow data is obtained from PCAP data [57, 58]; and when defining sessions originally [56, 59].

Gaikwad and Thool [54] proposed relevant features from the NSL-KDD dataset to improve classification accuracy and reduce the false positives rate. Potluri and Diedrich [55] claimed that DNNs have the capability of parallel computing, which makes them suitable for looking through large amounts of data with accelerated performance, making them a powerful tool for intrusion detection systems. Verkerken *et al.* [56] showed that an autoencoder yields the highest area under Receiver Operating Characteristics (AUROC), followed by one class support vector machine, isolation forest & principal components analysis, respectively. This outperforms previously proposed techniques when tested against the same evaluation set, CIC-IDS2017. The Netflow-based datasets generated by Sarhan *et al.* [57] were labeled to solve binary-class and multi-class classification tasks from four public datasets. Preliminary results using the generated Netflow-based datasets showed similar binary-class classification results as the four original datasets but lower multi-class classification compared to all of the original datasets. Engelen *et al.* [59] showed how data collection issues could significantly impact model evaluation, provided recommendations for anticipating and preventing such problems in the future, and stressed the importance of considering representativeness when developing high-quality benchmark datasets. They also illustrated how small changes made during the dataset preparation process could significantly improve machine learning benchmarks on the final dataset.

Finally, there are hybrid analysis studies that simultaneously handle both session-level and packet-level data. Hindy *et al.* [61] emphasized the importance of using flow-based features to discriminate between malicious and normal traffic, whereas packet-based features could suffice for traditional network attack detection needs.

5.1.3 Our Differentiation Strategy Compared to Related Work

Here, we gained an understanding of how current research on ML-based NIDS is being conducted. We found that most of the existing studies were conducted only using ML as a tool, and few studies comprehensively evaluated the practicality of ML as an NIDS. Also, although they claim high performance, they have only evaluated accuracy limited to given public datasets. The system is not guaranteed to work well even if the network environment or data changes. Furthermore, they often use only given features from public datasets or generalized features and lack ingenuity and consideration for features. We expect richer information, such as packet headers, payloads, and sessions, should be used in a hybrid manner.

We consider that ML-based NIDS should be studied by comprehensively discussing the following aspects.

- Diversity of known/unknown attacks in the training data and how to prepare for them
- Cost of analyzing alerts
- Real-time performance considering processing time and required hardware resources
- Learning by integrating features with rich information
- Categorization (multi-class classification) and explainability of alerts (root cause and severity of attacks)
- Generalization capabilities such as model re-training and transferability

Regarding the first item, how to prepare training data, we have proposed a technique to retransmit any traffic data with tcpreplay [62] and utilize alerts from multiple state-of-the-art NIDS to label the attack information [63,64]. Then, in this study, we proposed a methodology to reduce the cost of analyzing alerts by doing session-level classification. Our session-level approach is based on consolidating packet-level classification results rather than doing session-level classification directly with some session-level features. Our method is independent to how the packet-level classification is done. Therefore, when a novel packet-level classification approach is developed, we can use our session-level approach with it. The remaining aspects are

future work, and we plan to continue our research and development steadily toward the goal of a highly practical NIDS and a comprehensive security operations tool [65].

5.2 Methodology

In this section, we describe our approach for traffic classification. Firstly, we introduce the packet-level classification method. This method uses features extracted from each packet to determine whether the packet is benign or malicious. Secondly, we introduce our main contribution: the session-level traffic classification based on the packet-level classification. This approach consolidates outcomes of the packet-level classification to classify each session, where a session is defined as a collection of related packets. This approach can be used with arbitrary packet-level classification methods. Further, this method does not require designing new features extracted from sessions.

5.2.1 Packet Level Traffic Classification

Traffic monitoring is basically done by capturing packets. Therefore, using features extracted from each of captured packets is one of the natural ways. Traditional machine learning classifiers output their prediction y for a given feature vector x. In the binary classification case, y belongs to $\{0, 1\}$, and in the multiclass classification case, y belongs to some set of classes. In packet-level classification, each packet yields a feature vector to be input to classifiers. Therefore, classifiers generate predictions for each packet.

In more detail, many machine learning models for classification output their predictions in probabilistic form. In the binary classification case, a model outputs a real number in $[0, 1]$. We can interpret this as a probability of the event that the input sample is a positive sample. This value can also be interpreted as confidence in the event. In the multiclass classification case, a model outputs a real vector with the same dimension as the cardinality of the set of classes. The output vector has a constraint that all components are in $[0, 1]$, and their sum equals 1. Such a vector can be interpreted as a discrete probability distribution over the set

of classes. Each component of the vector is interpreted as a probability of the event that the input sample belongs to a corresponding class. When we have such a probabilistic output, we transform it into a prediction y described above. In the binary case, we apply some threshold to it. If the output probability is smaller than the threshold, we decide $y = 0$ otherwise, $y = 1$. In the multiclass case, we choose a class c whose corresponding component in the output vector is maximum, then we determine $y = c$ as a prediction. These probabilistic forms themselves also can be helpful as supplemental information in using predictions.

For packet-level classification, a classifier requires features extracted from each packet as its input. In this paper's experiments, we employ features used by Kitsune [28], an anomaly detection system. The Kitsune features are computed by processing packets successively. We introduce the Kitsune features below and discuss other features.

Kitsune Features

Kitsune [28] is a neural network-based anomaly detection system. Kitsune includes a packet-level feature extractor that focuses on frequencies or densities of traffic between the same hosts. Kitsune uses features extracted by its feature extractor for input to its anomaly detector. Kitsune's anomaly detector also contains some unique techniques. However, we can use the feature extractor independently. Though Kitsune features were designed for anomaly detection, they also can be used for packet classification and can achieve good performance [66].

A feature extractor of Kitsune has internal states called incremental statistics. To update the states, the feature extractor uses information from a captured packet. In concrete, it uses a timestamp, a packet size, MAC addresses, IP addresses and TCP/UDP ports related to them.

For more detail, an incremental statistic is a 3-tuple of real values denoted as $IS_\lambda = (w, LS, SS)$, where $\lambda > 0$ is a parameter to determine a decay rate of its time decay. The members of the incremental statistic (w, LS, SS) stand for 'weight', 'Linear Sum' and 'Squared Sum' respectively. The weight reflects the count for packets, and the linear/squared sum reflects the cumulative sum of (squared) values on which the incremental statistic focuses.

Each incremental statistic IS_λ is related to a data stream determined by MAC addresses, IP addresses, and TCP/UDP ports. Each packet is related to 4 data streams of different types.

- srcIP : an IP address of source of packet
- srcMAC-IP : (srcMAC, srcIP), a pair of MAC address and IP address of the source of a packet
- Channel : (srcIP, dstIP), a pair of srcIP and an IP address of the destination of a packet
- Socket : (srcIP, srcPort, dstIP, dstPort), a 4-tuple of IP addresses and TCP/UDP ports used by a packet

The Feature extractor updates these 4 incremental statistics for each packet. Each incremental statistic is initialized by zero values. The 3 types of incremental statistics except for the Channel-type focus on the frame length of a given packet. On the other hand, a Channel-type incremental statistic focuses on a jitter value defined as the difference between the timestamp of a given packet from the timestamp of the last packet transmitted with the same IP addresses. Let x denote the frame length or the jitter value of a given packet. For a packet with a timestamp t, each incremental statistic is updated using the following.

$$\gamma = 2^{-\lambda(t-t_{\text{last}})},$$
$$(w, LS, SS) \leftarrow (\gamma w + 1, \gamma LS + x, \gamma SS + x^2),$$

where t_{last} means the timestamp of the last packet related to the same stream and $\leftarrow$ mean updates of variables on the left side. These updates use only information on a given packet under processing except the last timestamp t_{last}. Therefore, it can be done successively without large memory consumption.

After the updating, we can calculate some scalar statistics based on updated (w, LS, SS). For srcIP and srcMAC-IP types, 3 kinds of statistics are defined. For Channel and Socket types, 7 kinds of statistics are defined. Therefore, we obtain 3 + 3 + 7 + 7 = 20 kinds of statistics from the incremental statics of 4 types.

The parameter λ of an incremental statistic takes a role to determine the intensity of time decay done by multiplying γ. The feature extractor uses multiple values of λ for a data stream. For each packet and value of λ, the extractor extracts 20 features described above. Finally, the extractor concatenates these features over the values of λ and outputs the concatenated vector as a vector for a packet. In [28], $\lambda = 5, 3, 1, 0.1, 0.01$ are employed. The extracted feature vectors are $5 \times 20 = 100$ dimensional vectors. In this paper, we use the same setting.

Kitsune features have a shortcoming in terms of their computation time. The longer the feature extractor is working, the longer its average execution time per packet is getting. This shortcoming is caused by increasing the number of statistics to be updated. In our experiments using EPYC 7702 64-Core Processor, the feature extractor could process about 290 packets per second in average for a PCAP file with about 64000 packets, and about 60 packets per second in average for a PCAP file with about 1140000 packets. However, this problem can be eased by limiting the number of statistics to be updated. When we limit the number of statistics, the required time to process a packet is bounded and a certain processing packet rate is guaranteed. However, in the experiments reported in this paper, we did not employ such modification.

Discussion: Other Packet-Level Features

The extraction algorithm for Kitsune features uses only timestamps and header information of packets; MAC addresses, IP addresses, port numbers, and frame lengths. The algorithm does not use the payload information of packets. Then, Kitsune features can be used even when we cannot access payloads. However, some kinds of anomalies will have important information in their payload of packets. To detect such kinds of anomalies, Kitsune features may not help. Another limitation of Kitsune features is a difficulty of parallelization of extraction. Because the extraction procedure depends on the successive processing packets in their temporal order.

Methods using payload information to detect anomalous packets are also argued. For example, Hassan et al. [29] introduced packet-level features based on payload information called payload embeddings. It employs Word2Vec [67], a famous word embedding technique used

in natural language processing. Word2Vec transforms each byte in a packet's payload into a fixed-length vector called a byte embedding. A payload embedding is the mean vector of byte embeddings of bytes in a packet's payload. Word2Vec model allows payload embeddings to extract contextual information of bytes in a payload. The payload embeddings can be calculated for multiple packets in parallel.

In contrast to Kitsune features which use only header information, payload embeddings use only payload information. Therefore, it may increase detection performance by combining both features. It is a future work.

5.2.2 Session Level Traffic Classification

We explained a packet-level classification method. Unfortunately, packet-level classification has some defects. First, in general, multiple packets are transmitted to achieve a purpose. For example, when we transmit some data using the transmission control protocol (TCP), the data are transmitted through a TCP connection. A TCP connection usually consists of multiple packets, including packets for connection establishment via the 3-way handshake and packets that contain partitioned data as their payload. Then, seeing only a single packet in such packets may not capture important characteristics in the traffic. Further, the number of captured packets is usually very large. When we classify each packet and report the result for them as packet-wise alerts, the number of reported alerts can also be too large. Too many and too fine alerts will be a heavy load for security operators to explore.

We define a session as a collection of packets transmitted to achieve the same purpose. Then, we propose an approach to convert packet-level predictions into session-level predictions. Our definition of sessions naturally reflects the property that a purpose of traffic is achieved by transmitting multiple packets. Further, the number of sessions is obviously smaller than the number of packets. Then, the number of reported session-level alerts will be suppressed. These properties allow us to overcome the defects of the packet-level traffic classification we described above.

When traffic is through a TCP connection, the TCP connection itself is a collection of

packets. However, other protocols do not have such an internal structure. Even if a traffic protocol is TCP, the whole connection is not always captured completely. Our definition of sessions can be applied to traffic regardless of their protocol and also to incomplete captures.

Definition of Sessions

We define a session as a set of packets that are transmitted through the same connection. In this definition, the connections are identified via IP addresses and port numbers used for the transmission. In concrete, packets transmitted between the same IP addresses and the same port numbers are members of the same session. We can also use the timestamps of packets to define sessions. For example, a packet that is transmitted after a large interval from the last arrival of the packet using the same IP addresses and ports can be regarded as an initial packet of a new session. If we employ this definition, a threshold for the interval has to be determined carefully. In this paper, we employ the simple definition without using timestamps. So, the session to which a given packet belongs is identified by an unordered pair of 2-tuple {(source IP address, source port number), (destination IP address, destination port number)}. Figure 5.1 explains the definition. The packets between the IP address X.X.X.X with port number x and the IP address Y.Y.Y.Y with port number y belong to the session s1. The session s2 consists of other packets between Y.Y.Y.Y:y and Z.Z.Z.Z:z.

Consolidating Packet Level Prediction to Session Level

We expect that session-level traffic classification will achieve more effective performance. However, in a natural way, we need to design session-level features used for the classification alternatively to packet-level features. Such design of features is not explicit. This paper proposes converting packet-level predictions to session-level predictions. We can do session-level classification without designing some session-level features.

Assume we already have packet-level predictions by some way. For example, we can obtain them by extracting Kitsune features and inputting them to some classifier. However, the approaches described here do not depend on how the packet-level predictions were obtained.

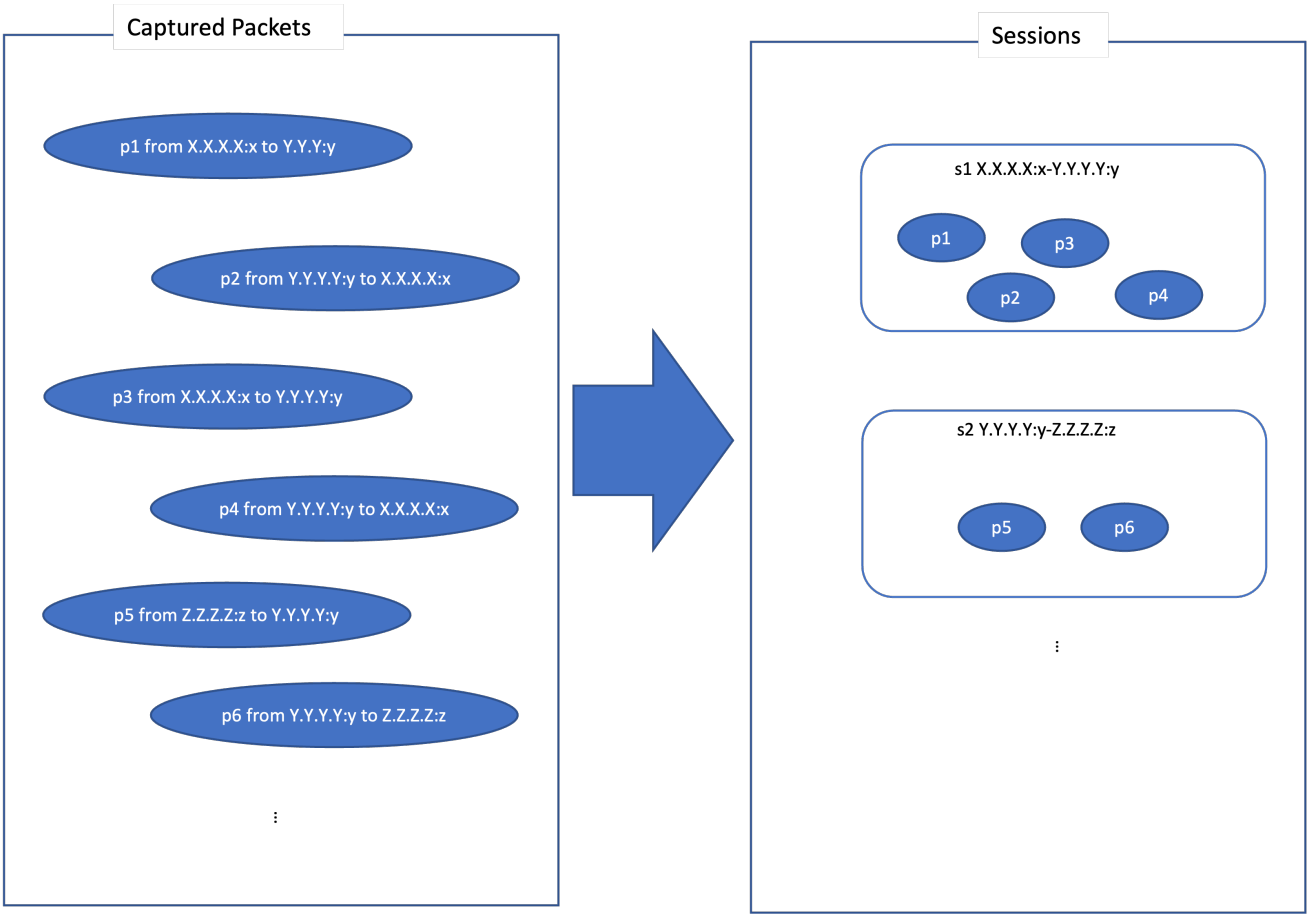

Figure 5.1: Explanation for the definition of session

We can use any features or techniques to obtain packet-level predictions.

The most simple way of converting packet-level predictions to session-level is taking inclusive disjunction over packet-level predictions in each session. In other words, we predict that a session is positive if and only if the session contains at least 1 packet predicted as positive. This approach depends on only the most suspicious packet in each session and can be regarded as assigning the session the maximum of predicted probabilities for packets in the session. This procedure realizes sensitive detection of suspicious sessions. Even if most of the packets in a malicious session do not include sufficient clues to detect them, there will often be a packet that can be detected in the session. In such a case, we can detect the malicious session via the converting procedure above. Figure 5.2 shows an explanation of this method. In this figure, the packet p3 is predicted as malicious by a packet-level classifier. Therefore, the session that contains it is predicted as malicious at the session-level. The other session consists of packets p4, p5, p6 does not contain a packet predicted as malicious. The session is predicted as benign. We call this simplest approach the basic approach in this paper.

The consolidating can be regarded as a binary classification problem that we classify each session based on the information extracted from the packet-level prediction results for the packets in the session. We can use an ML approach to solve this problem. In other words, we train a classifier that classifies sessions based on the packet-level prediction. In this approach, we design some features calculated from the output of the packet-level classifier. For example, the maximum, mean, or standard deviation of output (probability) in the sessions can be used. Note that if the maximum is directly used as the output of the classifier, this approach corresponds to the basic approach described above. This approach is a more flexible version of the basic approach and will balance the trade-off between evaluation metrics. In this paper's experiments, we compare this trainable approach to the basic approach. In summary, we examine the following two approaches for consolidating predictions.

- Basic approach: The maximum packet-level prediction among a session is used as a prediction for the session.
- Trainable approach: Some statistics are calculated for each session by using packet-

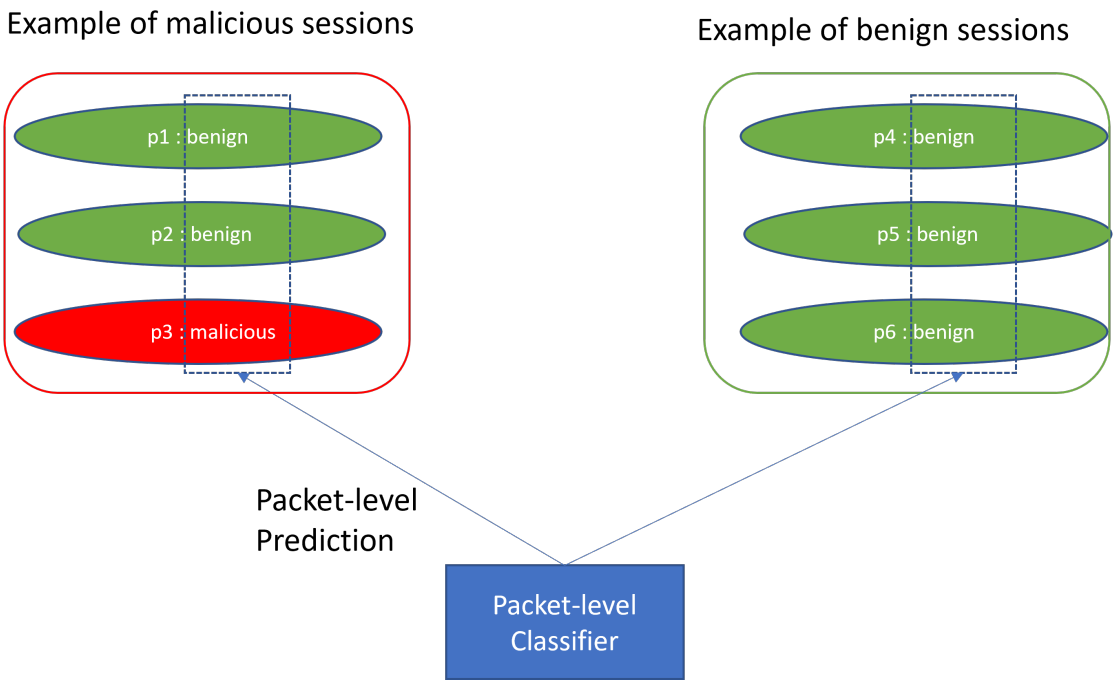

Figure 5.2: Explanation for the session-level classification method using packet-level prediction

level predictions among it. A trainable classifier classifies each session based on the calculated statistics.

Discussion: Other rules for consolidating

There are other rules to determine a prediction for each session using packet-level predictions. We can use the rate of the number of suspicious packets in each session. For example, if a given session consists of 10 packets and the number of suspicious packets in them is 2, the rate of the session is 0.2. Now, we output a real value in $[0, 1]$ for the session. This is the same situation as the binary classification with output in the probabilistic form. We apply a pre-determined threshold to this rate, for example, 0.5. In this case, this session contains suspicious packets. However, the rate is smaller than the threshold. Therefore, the session is predicted as benign. Note that the simplest rule described above is a special case of this rule using the rate. In other words, we can obtain the same result from these rules by using a sufficiently small threshold.

Using the number of detected packets is another choice. In this approach, we fix an integer threshold. When the number of detected packets in a session exceeds the threshold, the session is regarded as malicious. This approach is a natural extension of the basic approach. However, there is a shortcoming that the optimal threshold will depend on the size of each session. Therefore, we need to design a variable threshold mechanism based on session size.

Discussion: A merit of the basic approach

The basic approach is the simplest and trivial approach. This approach may lead to too sensitive detection and low precision. The trainable approach may overcome this shortcoming. However, the basic approach has a property that it can output a conclusion in the middle of a session. If we detect a malicious packet in the middle of a session, we can conclude the session is malicious based on the basic approach. This realizes the most quick detection. Further, we need not to process the remaining packets in the session. Other approaches which refers whole of a session including the trainable approach will not have these property. The approach based

on the detected packets discussed above also has this property. However, when we employ a variable threshold based on session size, this property will be lost.

The results of our experiments, which are shown later, show that the basic approach can achieve sufficiently high performance when we have a good packet-level classifier, and we can enjoy the merit described above without loosing the performance significantly. Therefore the basic approach can be a practical choice in terms of both performance of classification and the number of packet to be processed.

Discussion: Multiclass Classification as A Future Work

In this paper, we focus on binary classification methods. However, if the packet-level classification method can be used in the multiclass case, the session-level classification will be possible to be done in the multiclass form. For example, the majority voting in each session can be done by using the outcomes from the packet-level multiclass classification method. Using the mean value of the packet-level outcomes will also be hopeful. These extensions to the multiclass classification are future work.

5.3 Experiments

In this paper, we show the experimental results of the binary classification method we proposed.

5.3.1 Dataset

Since our method executes packet-level prediction, we need a dataset which provides packet-level samples and labels. Unfortunately, to the best of our knowledge, there are no datasets which provide explicitly packet-level labels. Existing labeled datasets are basically provided as a set of statistical features with labels and the features were extracted from a flow, which is a sequence of packets defined by a tool used for generating the dataset. Even when

packet-level samples are included in a dataset, they are only an auxiliary material without labels. Further, the flow-level features usually do not contain sufficient information to identify which packets are a member of the flow. Therefore, we cannot use the provided labels directly for evaluations of methods which require packet-level processing. However, some datasets provides meta information which help us to obtain packet-level labels.

We use an open dataset CSE-CIC-IDS2018 [46]. This dataset provides a set of PCAP files and meta information on attacks that the PCAP files contain. The PCAP files were separated based on their date of capture and the IP address of the host machines. This dataset contains PCAP files for 10 days. Our experiments employ the 7 days of these 10 days except for 02/21, 03/01, and 02/28. The PCAP files of the victim IP on 02/21 are too large (20GB over) for our environment. The PCAP files of the victim IP on 02/28 and 03/01 contain only 2 and 3 malicious sessions, respectively. Then, our data splitting process for experiments described later will not work properly for the data of these days.

5.3.2 Preprocessing

To use the PCAP files in the dataset for our experiments, we need some preprocessing.

Merging PCAP Files

Most of the attacks that appear in this dataset targeted a single IP address. Then all traffic of each attack belongs to the PCAP file(s) related to the victim's IP address. We call such PCAP files 'main PCAP files'. The packets in the main PCAP files are always related to the victim's IP address. To make various IP addresses appear in the data we use, we merge other PCAP files on the same day to the main PCAP files. We call these PCAP files 'additional PCAP files.' The choice of additional PCAP files is done by the following procedure for each day. First, we sort the PCAP files on the day except for the main PCAP files in ascending order of their file size. Second, we employ the PCAP files from the head until the sum of file sizes of employed files exceeds the total file size of the main PCAP files. Finally, we merge all PCAP files in the main PCAP files and employ PCAP files. In other words, we generate a new PCAP

file that contains all packets in the main and the additional PCAP files. The packets are sorted in ascending order of their timestamp.

Labeling Data

In the experiments in this paper, we employ a classifier based on supervised learning. Supervised learning approaches require labeled data for their training. Many datasets including CSE-CIC-IDS2018 usually do not provide packet-wise labels. However, many datasets provide meta-information on attacks. For example, attack names, duration, and IP addresses of victims or attackers are provided. For the experiments, we generate packet-wise labels by using such information. In concrete, packets that match the information of an attack are labeled the attack name. The others are labeled benign. For binary classification, we convert benign labels into negative and attack names into positive.

In our experiments using CSE-CIC-IDS2018, we use meta-information on the duration and IP addresses of attacks. For example, the dataset provides meta-information that from '2018/02/14 10:32' to '2018/02/14 12:09', an attack 'FTP-BruteForce' was executed by the attacker with IP address '18.221.219.4' and the victim's IP address was '172.31.69.25'. We can label packets in 2018/02/14 using this information. In concrete, the packets transmitted in the duration 2018/02/14 10:32 to 2018/02/14 12:09 between IP addresses 18.221.219.4 and 172.31.69.25 are labeled 'FTP-BruteForce'. Meta-information on other attacks is also provided. Packets not labeled any attack name by this procedure are treated as packets of benign traffic and labeled 'BENIGN'.

Splitting Data

The dataset is provided in the PCAP format. Unfortunately, the dataset is not separated in the train/test format. Generally, for fair evaluation, it is preferred that data used for the training and the testing phases are independently prepared. For some techniques like neural networks, data used for validation are also preferred. It can be used to avoid overfitting.

To generate data in train/test format from a single data, random shuffling, and sampling

to split data sets are often done for machine learning settings. Also, researches on NIDS [27, 29] usually employ this standard method to split data. However, this splitting lacks some concern for a property of packet capture data. In the case of packet capture data, each packet will have relationships with nearby packets. Then, simply shuffling and sampling packets may cause the leakage problem that the training data include samples implicitly containing the answer for the testing data. To avoid this problem, we split data at the session level. First, we assign each session to one of the train/validation/test data. Then, we split the PCAP file into 3 PCAP files corresponding to train/validation/test, respectively. So, we have the split data in the PCAP format. The assignment for sessions can be done randomly. However, for the reason of reproducibility, in this paper, we assign sessions periodically in the respective classes. In concrete, we assign the first session with class "BENIGN" to the train set, the second to the validation set, the third to the test set, the fourth to the train set, and so on. Sessions with other classes are also assigned in the same way, respectively. The proportion of each class in each subset becomes about the same. This session level splitting approach can help models to avoid the leakage problem caused by using packets from the same session as packets in a test set. This will lead to fairer evaluation. A possible shortcoming of this methods is that this method will be affected by the difference of numbers of packets of each session. When there are sessions with large number of packets and with small number of packets, This method may cause the split sets with imbalanced packet numbers. Further, such sessions may have different natures. When the number of sessions is small, these concerns may appear explicitly.

Over-Sampling

The number of packets related to some attacks is usually less than the number of benign packets. In other words, many datasets of the captured traffic are imbalanced data. Training using imbalanced data causes poor performance of trained models. To overcome this problem, over-sampling techniques are often used. In our experiments, we employ an over-sampling technique SMOTE [68, 69], which synthesizes samples with minority classes using neighboring samples.

5.3.3 Detail of Experiments

Our experiments were done for data each day. We employed 3 kinds of binary classifier, feed-forward neural network(FFNN), XGBoost [70] Kitsune anomaly detector [28].

The FFNN is also known as multi layer perceptron and is one of basic methods for ML-based NIDSs. The XGBoost is an open source system which provides effective implementations of algorithms based on gradient tree boosting [71]. The XGBoost is known as a popular choice of ensemble method for various application of machine learning. The Kitsune anomaly detector is the original usage of features we used. This is an anomaly detector rather than a binary classifier. It outputs anomaly scores which belong to $(0, \infty)$, which are based on reconstruction errors of auto-encoders. However, we can use it for binary classification by setting an appropriate threshold for anomaly scores.

In each experiment, we executed the preprocessing described above for the data of the day. Then, Kitsune feature extraction was executed for each split of data. After the extraction, a classifier was trained by using train/validation data, where the validation data was used for the early stopping technique. After the training, we obtained predictions for packets in the test data by the trained classifier. We converted the packet-level prediction to the session-level prediction. Finally, we evaluated the predictions.

We implemented FFNN classifiers by using the TensorFlow framework. Packet-level FFNN classifiers used in our experiments had 3 hidden layers, and each of them had 64 ReLU units. Batch normalization layers also were put after each hidden layer. In training, Adam optimizers were used with an initial learning rate of 0.001. The early stopping patience parameter, which determines the timing of early stopping, was set to 10. When the early stopping work, the weights of the classifier, which minimize the loss for the validation data, were restored.

We used XGBoost classifiers with default setting in XGBoost Python package version 1.6.2 except for early_stopping_rounds parameter was set to be 10. This parameter takes a similar role to the patience parameter for FFNN classifiers.

For experiments using a Kitsune anomaly detector, we used the reference implementation available at Github [72]. Since Kitsune is an anomaly detector based on auto-encoders [28],

we used only benign samples in the train sets to train the models. The parameters were set to be the same as the example code in [72] except for the number of samples used to train the anomaly detector. The default setting uses a limited number of samples to train the anomaly detector. However, we used all benign samples in the train set.

To execute binary classification based on outputs from a trained classifier, we need to select a threshold to be applied to them. For a given packet, when a classifier outputs a value smaller than the threshold, we consider the packet benign. Otherwise, we consider the packet malicious. The FFNN and XGBoost classifiers output a value which can be regarded as a probability. Therefore, we employed 0.5 as the threshold for them. In contrast, the output of Kitsune is not a probability but an anomaly score. Therefore, there are not a natural threshold like 0.5 for a probability. We employed the 95% quantiles of the anomaly scores for the benign samples used for the training as a threshold for the anomaly scores for executing binary classification based on them.

To obtain session-level predictions, we used the basic approach and the trainable approach to convert predictions. In the basic approach, a score of a session is the maximum of packet-level outputs of classifiers in the session. In the trainable approach, we defined a 6 dimensional session-level feature that consists of the session_size, the positive_count, the positive_rate (positive_count / session_size), the maximum of the outputs, the mean value of the outputs, and the standard deviation of the outputs. In this definition, the session_size is the number of packets consisting of the session, the positive_count is the number of the packets predicted as malicious. Using these features as input, we trained a session-level classifier. For packet-level FFNN classifiers, we used a FFNN with a single hidden layer with 128 ReLU units followed by a batch normalization layer. For packet-level XGBoost classifiers and Kitsune anomaly detectors, we used an XGBoost classifier with the same setting as the packet-level.

The training data and the validation data is originally the same as the training data for packet-level classifier. However, we converted them into session-level based on the same procedure as the basic approach. Settings for the optimization is same as for the packet-level classifiers.

In the experiments using FFNN classifiers, the training and evaluation steps, including the

initialization of weights of a classifier, were executed 10 times. This is because the performance of neural network-based classifiers generally depends on their weights that are initialized randomly. The performance measures we show in the next subsection are the mean value of the results of the 10-time evaluations. In the experiments using XGBoost classifiers and Kitsune anomaly detectors, we executed these steps only 1 time. Because, the XGBoost algorithm with its default setting is deterministic, and the reference implementation of Kitsune we used fixed the random seed so that its behavior is also deterministic.

5.3.4 Results of Experiments

We employed the following measures for evaluation, 'True Positive(TP)', 'False Positive(FP)', 'True Negative(TN)', 'False Negative(FN)', 'Accuracy', 'Precision', 'Recall(a.k.a. True Positive Rate)', 'True Negative Rate(TNR)', and 'F1-measure(F1)'. The TP is the number of positive(malicious) samples predicted as malicious correctly, and the FP is the number of negative samples predicted as malicious. Conversely, the TN is the number of negative samples predicted as benign correctly, and the FN is the number of positive samples predicted as benign. Other measures are defined by these numbers as follows.

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\
\text{TNR} &= \frac{\text{TN}}{\text{TN} + \text{FP}} \\
\text{F}_1 &= 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\end{aligned}$$

We also employed 'Area Under the Curve of Receiver Operating Characteristic curve (ROCAUC)'. Only ROCAUC is independent to the threshold.

Note that in our results by FFNN classifiers shown later, all metrics are a mean value of 10-time experiments. This is because TP, FP, TN, and FN are not integers in the results.

Results of Packet Level Traffic Classification

Table 5.2 shows the measures of the packet-level predictions of our experiments using FFNN classifiers, XGBoost classifiers, and Kitsune anomaly detectors.

In the experiments, packet-level classification using XGBoost classifiers seems to be the best. It achieved F1-measures exceeding 98.5% for all days. FFNN classifiers achieved F1-measure of about 98% at most and notably poor F1-measure for 02/23 and 03/02, about 62% and 76% respectively. The results for Kitsune anomaly detectors seem to be the worst. Their Precisions are low for many days, and their Recalls are extremely low for 02/14 and 02/23. Their F1-measures are at most of about 84%.

These measures can be improved if we can select more appropriate threshold. However, the ROCAUCs, which are independent to the selection of a threshold, were also achieved the best values by XGBoost classifiers.

Table 5.3 shows the number of packets (n_samples) and the number of packets predicted as malicious correctly (n_detection) for each malicious class in a day using FFNN classifiers, XGBoost classifiers, and Kitsune anomaly detectors. It also shows class-wise Recall defined as n_detection / n_samples. Note that class-wise Precision and F1-measure cannot be defined for our experiments since our experiments are of binary classification. It is shown that FFNN classifiers and XGBoost classifiers tended to overlook attacks labeled SQL Injection. Recalls of other attacks were high. Especially, the XGBoost classifiers achieved Recalls exceeding 99% for all other attacks. It is also shown that Kitsune anomaly detectors tended to overlook many kinds of attacks, even though other classifiers detected them well.

The numbers of packets labeled SQL Injection are especially small. This might cause insufficient fitting even though we used oversampling technique. The traffic of BruteForce-Web and SQL Injection are included in the data from 02/22 and 02/23. In Table 5.3, the Recall achieved by FFNN classifiers for BruteForce-Web on 02/22 is about 99%. However, the Recall for the same kind of attack on 02/23 is about 21%. In the table, the Recall achieved by XGBoost classifiers for SQL Injection are about 17% and 94% on 02/22 and 02/23 respectively. Such difference in Recall implies that the characteristics of a kind of attack can vary in

other data.

There is a possibility that we can improve performance by using other classifiers. For example, classifiers using payload information of packets will be effective for attacks whose characteristics are in their payload. The SQL Injection will be a kind of such attack.

Results of Session Level Traffic Classification

Table 5.4 and 5.5 show the measures of the session-level predictions achieved by using each classifier. These results show that the trainable approach tended to suppress the number of FPs and to achieve better Precision than the basic approach for each classifier. Further, when we used XGBoost and Kitsune anomaly detectors, the trainable approach tended to achieve better F1-measures than the basic approach.

However, when we used XGBoost classifiers, the difference between the basic and the trainable approach was relatively small. Indeed, using XGBoost classifiers with the basic approach achieved sufficiently good performance. In particular, their Recall was saturated and using the trainable approach did not improve it. As we described above, the basic approach has some advantage caused by its simplicity. Therefore, using XGBoost classifiers with the basic approach will be an attractive choice. We discuss the advantage of the basic approach based on the prediction by XGBoost classifiers in the next subsection.

We also provide results on class-wise Recall at the session level. Table 5.6 shows the number of sessions (n_samples) and the number of sessions predicted as malicious (n_detected) for each malicious class in a day. It is shown that using XGBoost classifiers achieved detecting most of the malicious session with both the basic and the trainable approach.

As well as the packet-level results, all approaches suffered from low Recalls for SQL injection class. The small number of samples of this class may cause this problem. There is another possible reason for this problem. That is the property of the class and the features we used. Attacks of SQL injection is done by sending requests which contain some malicious SQL statements. Therefore, essentially, important footprints of this kind of attack will appear in payloads. However, Kitsune features we used in the experiments do not refer to payloads

of input packets. These facts may be the reason for the poor performance. Using other kinds of features will help us ease this problem. For example, Iida et al. [73] proposed to use both Kitsune features and payload embeddings [29] with experimental results in which high Recalls were achieved for SQL injection.

An Advantage of the Basic Approach

When we use the basic approach, we can operate the session-level prediction by using only the some number of first packets in each session.

We simulated the situation using the packet-level prediction results by XGBoost classifiers shown above. In concrete, we processed successively process the packet-level predictions in the temporal order of packets. When we found a packet predicted as malicious, we concluded the session to which the packet belongs was malicious and ignored the rest of packets belonging to the same session. For a fixed natural number k, if all of first k packets belonging to a session were predicted as benign, we concluded the session was benign and also ignored the rest of packets. We call this procedure as first k prediction procedure. We executed simulations of the first k prediction using $k = 1, 5, 10, 15, \infty$. The value $k = \infty$ means that when all packets in a session were predicted as benign, we needed to process them all to conclude the session was benign. This setting is a baseline.

Table 5.7 shows the results of this simulation using all samples of sessions in the all days. The n_processed column presents the average number of packets processed to conclude a session was benign/malicious. Each row presents results using k described. Table 5.8 shows class-wise Recalls and average numbers of processed packets in each session.

Table 5.7 implies that we could achieve good performance of session-level classification by using only a few packets from a head of each session. The n_processed for the baseline ($k = \infty$) was about 17. In contrast, it for the $k = 15$ simulation was about 9. Interestingly, even when we used only the first packet of each session ($k = 1$), the better F1-measure than the baseline was achieved. The Recall decreased from the baseline, however, the Precision increased. The improvement of the Precision and the F1-measure could be observed for all k. This means that

packets of head parts of many FP sessions were classified correctly. Also the Table 5.8 implies that most of attacks in the dataset could be detected using only the first packet of each session.

In conclusion, the first k prediction procedure could achieve good session-level classification efficiently in terms of the number of packets to be processed by a classifier. In our environment with AMD EPYC 7702 64-Core Processor, XGBoost classifiers took 0.0026 seconds to classify each packet in average. Since the difference in n_processed between $k = 1$ and $k = \infty$ was about 16, using $k = 1$ saved about 0.04 seconds in classification a session in average. Using $k = 5$ which maximized the F1-measure also saved about 0.03 seconds.

Reduction of the number of targets

Table 5.9 shows the number of packets (n_packets), the number of sessions (n_sessions), and the ratio of them for each class and each day in the dataset used in our experiments before train/validation/test splitting. This shows that the numbers of sessions are smaller than 10% of the number of packets in the dataset. The Total row in the table shows the numbers of packets and sessions of all the classes and all the days, and the ratio of them. It is shown that the total number of sessions is about 5.9% of the number of packets in whole of the data we used. Then the conversion from the packet-level to the session-level could decrease the number of detections, as also shown in Tables 5.2 – 5.6. Table 5.10 shows the numbers of malicious packets predicted as malicious, the numbers of sessions predicted as malicious , and the ratio of them using each classifier. When we issue alerts corresponding to a target(packet or session) predicted as malicious, these numbers correspond to the number of alerts. In our experiments regardless of kinds of classifiers and the basic/trainable approaches, the numbers of detected sessions of each class were less than about 10% of the number of detected packets for most of days. The Total row in the tables show that our methods using FFNN and XGBoost classifiers achieved reducing the number of detection to less than 2.5% of the number of the packet-level detection in whole of the data we used. Our method using Kitsune anomaly detectors also achieved reducing the number to less than 10%. These results imply that the session-level prediction will lead to a more organized alert set than the packet-level prediction.

5.4 Comparison of Experimental Results

We want to provide comparisons of metrics between our approaches and other approaches reported in related researches. However, almost all existing researches use the CSE-CIC-IDS2018 dataset as a flow-level dataset. They executed the classification of flow samples using extracted features defined by the generation tool used for the dataset. Such experiments are essentially different from our experiments using packet-level features extracted from the raw packet samples to classify session samples.

The most critical difference is between the definitions of our session and the flow. Basically, both are defined as a set of packets transmitted with same IP addresses and ports. Indeed, our session is just defined as this. However, the definition of the flow seems to involve timestamps of the packets and consider the notion of time-out. Therefore, a session may be a union of multiple flows. Unfortunately, details related on the time-out could not be found and flow samples in the dataset do not contains sufficient information to identify which packets are related to them.

The procedure for splitting data to separate a test set is also different. As described above, we executed the splitting with the consideration on the leakage problem. Most of the related works do not describe the procedure for data splitting in detail. Their procedure is probably the random splitting, which is the most standard one without the consideration on the leakage.

For such reasons, it is difficult to compare our results to results in the existing researches in the truly fair manner. In this paper, we compare our results to results reported in some related works. However, this is just for reference.

5.4.1 Comparison to Results of References

We cited experimental results from 18 related works. The works were found in a survey paper [74] on works using the CSE-CIC-IDS2018 dataset. Table 5.11 shows session/flow level metrics(Accuracy, Precision, Recall, F1-measure) for the CSE-CIC-IDS2018 dataset achieved by related works and our approaches with good performance using XGBoost classifiers with the basic approach and first $k(k = \infty, 1, 5)$ prediction procedure. The referential values in the

table were cited from Table 2 of [74] except for F1-measure. The F1-measures were calculated from the Precisions and Recalls.

In our experiments whose results shown in Table 5.7, the $k = \infty$ was a base line and achieved the best TP and FN, the $k = 1$ achieved the best FP, TN and Precision, and the $k = 5$ achieved the best F1-measure. This is why we employed these k in the comparison.

The cited values are the best respective values reported in each original paper. Note that some papers evaluated their results in class-wise manner and the cited evaluation metrics of them were for a certain class.

Some related works [75–78] achieved a complete classification for the dataset, whose metrics were 100%. The evaluations in [75] were executed class-wise and their performance for BruteForce-Web/XSS, and SQL Injection were not good (F1-measures were at most 51%). The related work [76] focused on detection of DoS and DDoS attacks and their experiments used only the part of the dataset including DoS/DDoS attacks. The related work [77] did not provide details on their experiments. However, it seems to have focused on detection of Botnet attacks and used only the part of the dataset including them.

Except for such works, our approaches overperformed most of the related works in terms of Accuracy and F1-measure. Therefore, our approaches were sufficiently competitive in terms of the classification results. Especially, the approach with $k = 1$ and $k = 5$ also has an advantage in terms of the quickness of the detection.

5.5 Discussion on Future Works

In this paper, our experiments are done by using a public dataset CSE-CIC-IDS2018 and a packet-level classification method that uses Kitsune features and a feed-forward neural-net classifier. There are some future works to evaluate our session-level classification method.

5.5.1 Training using other datasets

As shown in Table 5.1, there are other public datasets used to evaluate ML-based NIDSs. To use various public datasets and compare the performance of our method for them to other NIDSs is future work.

Using real traffic data

Further, in practice, we will want to use some real traffic data to adapt models to real attacks. Evaluating our method using real traffic data is also future work. In our experiments, we used a public dataset providing meta-information on the contained attacks. Then, we could use the information to label traffic for the training. To use real traffic data for training models, we require information on attacks in the traffic. One of the reasonable ways to obtain such information is to use alerts raised by existing NIDSs [63, 64]. An alert from an NIDS provides us with information to identify packets related to the alert and the kind of anomaly related to the alert. We can label packets in real traffic data by using such information. The performance and the characteristics of models trained with this approach will depend on what kind of NIDS we used. We can use multiple kinds of NIDSs to label the data. A model trained with such labels will have a characteristic that integrates the characteristics of the NIDSs we used.

5.5.2 Using other classifiers

In our experiments, we used FFNN, XGBoost, and Kitsune as a classifier for packet-level classification. The many classifiers used in the related works in Table 5.1 can also be used with our approach. Also, the Kitsune features can be replaced or combined with other features. Exploring a combination of a classifier and features that allow our method to achieve better performance is future work.

5.5.3 Fair Comparison to Other Methods

We provided a comparison of experimental results of our session level classification to the related works. However as described above, the comparison is not in a fair manner for the reason of the difference in the dataset usage and the definition of sessions and flows. To compare our approach to related works in a fair manner, conducting experiments based on the same dataset usage and a modified definition of sessions is required. Such additional experiments and evaluation are future works.

5.6 Conclusion

The proposed approach for session-level analysis in network intrusion detection addresses the limitations of packet-level analysis, resulting in improved classification performance.By consolidating packet-level classification outputs obtained by using XGBoost classifiers, this approach reduces the number of false positive alerts, making it easier for security teams to manage and prioritize threats. The effectiveness of this approach is demonstrated through experiments on public benchmark datasets, which achieved an F1-measure of about 99%. The experimental results also showed that our approach can reduce the number of packets to be processed by a classifier without significant loss of classification performance. This research highlights the potential of machine learning-based network intrusion detection to adapt to evolving cyber threats and provides a foundation for future research aimed at improving the accuracy and efficiency of NIDSs.

Table 5.1: Summary of the recent related studies of ML-based NIDS

Data Type	Paper, Year	Dataset	Model	Result and Contribution
Packet – Header	Mirsky [28], 2018	NSL-KDD, CIC-IDS 2017, UNSW-NB15	AE	i) Plug & Play deployment ii) Unsupervised learning iii) Online processing iv) Low resource requirements v) Efficient feature extraction framework
	Hwang [47], 2019	ISCX 2012, USTC-TFC 2016	CNN, LSTM	Proposing a word embedding mechanism to extract packet semantic meaning; adopting LSTM model for learning temporal relation between fields in the packets headers
	Gwon [48], 2019	UNSW-NB15	LSTM, SVM	Utilization of both time series and categorial info along with LSTM-based approach for IDS
	Shrestha [49], 2021	CIC-IDS 2018	DT, NB, SGD, K-Means	Combined with 5G technology integrated into unmanned aerial vehicles (UAVs) to detect vulnerabilities & cyberattacks effectively
	Verkerken [50] 2021	CIC-IDS 2017, CIC-IDS 2018	AE, SVM, IF, PCA	Proposed an inter-dataset evaluation strategy that could be used in future research as well as real-world applications
Packet – Payload	Park [51], 2018	Kyoto 2006+	RF	Constructing new datasets from existing ones which can be further utilized for training purposes while building IDS
	Fang [52], 2020	DARPA98	ENN with SVM	Clustering algorithm employed by ENN largely reducing missing text info; SVM eliminating negative impact caused due to noisy data, thus improving overall performance
	Goodman [53], 2020	DARPA 2009	Word2Vec with SVM	Application of word embedding technique, Word2Vec, achieved a high area under curve (AUC) value
	Hassan [29], 2021	CIC-IDS 2017, ISCX 2012, UNSW-NB15, CTU13	SNN with k-NN	Shallow neural networks were employed to generate vector representations for bytes and their corresponding payloads and achieved high accuracy in intrusion detection
Session	Gaikwad [54], 2015	NSL-KDD	REPTree	Ensemble method of machine learning with Bagging and REPTree base class provides high classification accuracy, low FPR, and fast model building time
	Potluri [55], 2016	NSL-KDD	DNN	DNN can be effectively utilized for IDS with its parallel computing capabilities that allow them to look through large amounts of data quickly and accurately
	Verkerken [56], 2020	CIC-IDS 2017	AE, SVM, IF, PCA	AE yields the highest area under Receiver Operating Characteristics (AUROC), followed by SVM, IF, and PCA, respectively
	Sarhan [57], 2021	UNSW-NB15, BoT-IoT, ToN-IoT, CIC–IDS 2018	SVM, RF, DT	NetFlow features extracted from packet capture files of four benchmark NIDS datasets
	Corsini [58], 2021	CIC-IDS 2017, CTU13	LSTM, FFNN	LSTM achieved comparable performance with FFNN in the CIC-IDS 2017 dataset while obtaining superior performance on the CTU13 dataset
	Engelen [59], 2021	CIC-IDS 2017	RF	Uncovering a series of problems related to traffic generation, flow construction, feature extraction, and labeling; investigation into causes behind these shortcomings
Hybrid	Almseidin [60], 2017	KDD99	J48, RF, DT, MLP, NB	Evaluation focused on increasing detection rates by reducing false positives and false negatives, resulting in DT having the lowest false negatives and RF having the highest average accuracy.
	Hindy [61], 2021	MQTT	NB, DT, RF, SVM, k-NN	Demonstrated that the ML model can be suitably employed for IDS requirements for networks using the MQTT protocol and published the generated dataset.

	FFNN									
	TP	FP	TN	FN	Accuracy	Precision	Recall	TNR	F1	ROCAUC
02/14	1374640.2	24404.9	1192225.1	37627.8	0.9764	0.9826	0.9734	0.9799	0.9779	0.9910
02/15	96766.1	**1331.5**	**296503.5**	22679.9	0.9425	0.9875	0.8101	**0.9955**	0.8824	0.9915
02/16	70240.7	6314.0	2081636.0	121.3	0.9970	0.9186	0.9983	0.9970	0.9565	0.9980
02/22	9505.6	257.4	54165.6	217.4	0.9926	0.9736	0.9776	0.9953	0.9756	0.9957
02/23	6477.9	206.9	64127.1	6516.1	0.9131	0.9571	0.4985	0.9968	0.6174	0.9877
03/02	220767.0	27934.8	783964.2	107144.0	0.8815	0.8933	0.6733	0.9656	0.7635	0.9606
	XGBoost									
	TP	FP	TN	FN	Accuracy	Precision	Recall	TNR	F1	ROCAUC
02/14	**1407076**	10608	1206022	**5192**	**0.9940**	**0.9925**	**0.9963**	0.9913	**0.9944**	**0.9997**
02/15	119018	1359	296476	428	**0.9957**	**0.9887**	0.9964	0.9954	**0.9925**	**0.9996**
02/16	**70352**	**15**	**2087935**	**10**	**1.0000**	**0.9998**	**0.9999**	**1.0000**	**0.9998**	**1.0000**
02/22	9648	**212**	**54211**	75	**0.9955**	**0.9785**	0.9923	**0.9961**	**0.9853**	**0.9966**
02/23	**12961**	**112**	**64222**	**33**	**0.9981**	**0.9914**	**0.9975**	**0.9983**	**0.9944**	**0.9995**
03/02	**326892**	**7404**	**804495**	**1019**	**0.9926**	**0.9779**	**0.9969**	**0.9909**	**0.9873**	**0.9996**
	KitsuneAD									
	TP	FP	TN	FN	Accuracy	Precision	Recall	TNR	F1	ROCAUC
02/14	39601	**6345**	**1210285**	1372667	0.4754	0.8619	0.0280	**0.9948**	0.0543	0.9150
02/15	**119238**	45840	251995	**208**	0.8896	0.7223	**0.9983**	0.8461	0.8382	0.9320
02/16	69940	142273	1945677	422	0.9339	0.3296	0.9940	0.9319	0.4950	0.9492
02/22	**9680**	8402	46021	**43**	0.8683	0.5353	**0.9956**	0.8456	0.6963	0.9575
02/23	1953	4926	59408	11041	0.7935	0.2839	0.1503	0.9234	0.1965	0.8858
03/02	180570	40690	771209	147341	0.8350	0.8161	0.5507	0.9499	0.6576	0.8288

Table 5.2: Results of packet-level classification. The bold values mean the best values of each metric for each day among the kinds of classifiers.

		FFNN		XGBoost		KitsuneAD	
	n_samples	n_detected	Recall	n_detected	Recall	n_detected	Recall
SSH-BruteForce(02/14)	1412268	1374640.2	0.9734	**1407076**	**0.9963**	39601	0.0280
DoS-GoldenEye(02/15)	86017	73801.9	0.8580	**85830**	**0.9978**	85995	0.0280
DoS-Slowloris(02/15)	33429	22964.2	0.6870	33188	0.9928	**33243**	**0.9944**
DoS-SlowHTTPTest(02/16)	70362	70240.7	0.9983	**70352**	**0.9999**	69940	0.9940
BruteForce-Web(02/22)	6078	5991.4	0.9858	6060	0.9970	**6067**	**0.9982**
BruteForce-XSS(02/22)	3592	3504.1	0.9755	3579	0.9964	**3592**	**1.0000**
SQL Injection(02/22)	53	10.1	0.1906	9	0.1698	**21**	**0.3962**
BruteForce-Web(02/23)	5446	1167.7	0.2144	**5428**	**0.9967**	10	0.0018
BruteForce-XSS(02/23)	7424	5298.5	0.7137	**7416**	**0.9989**	1943	0.2617
SQL Injection(02/23)	124	11.7	0.0944	**117**	**0.9435**	0	0.0000
Bot(03/02)	327911	220767.0	0.6733	**326892**	**0.9969**	180570	0.5507

Table 5.3: Number of packets in each class and the class-wise results of packet-level classification. The bold values mean the best values for each class among the kinds of classifiers.

	FFNN							
	Basic				Trainable			
	TP	FP	TN	FN	TP	FP	TN	FN
02/14	4705.0	20.7	93368.3	10.0	4705.0	**0.0**	**93389.0**	10.0
02/15	6488.3	**114.7**	**25224.3**	542.7	**6611.5**	2322.4	23016.6	**419.5**
02/16	**4705.0**	176.5	159999.5	**0.0**	4704.7	**1.0**	**160175.0**	0.3
02/22	**61.8**	21.9	4502.1	**2.2**	55.9	**0.5**	**4523.5**	8.1
02/23	**64.3**	12.9	5933.1	**10.7**	26.0	**2.6**	**5943.4**	49.0
03/02	**23429.2**	3139.1	41259.9	**7131.8**	21609.1	**1392.2**	**43006.8**	8951.9
	XGBoost							
	Basic				Trainable			
	TP	FP	TN	FN	TP	FP	TN	FN
02/14	4705	1	93388	10	4705	1	93388	10
02/15	7031	103	25236	0	7031	**100**	**25239**	0
02/16	4705	1	160175	0	4705	**0**	**160176**	0
02/22	**61**	4	4520	**3**	60	**0**	**4524**	4
02/23	75	7	5939	0	75	**0**	**5946**	0
03/02	**30545**	861	43538	**16**	30482	**572**	**43827**	79
	KitsuneAD							
	Basic				Trainable			
	TP	FP	TN	FN	TP	FP	TN	FN
02/14	1405	1554	91835	3310	**4705**	**2**	**93387**	**10**
02/15	**7031**	2864	22475	**0**	6960	**79**	**25260**	71
02/16	**4705**	13942	146234	**0**	0	**0**	**160176**	4705
02/22	**62**	1318	3206	**2**	12	**3**	**4521**	52
02/23	32	398	5548	43	**62**	**1**	**5945**	**13**
03/02	20811	2616	41783	9750	**28237**	**1765**	**42634**	**2324**

Table 5.4: Results of session-level classification for TP, FP, TN, and FN. The Basic column shows results of the basic approach and the Trainable column shows results of the trainable approach. The bold values mean that the values are better than that achieved by the opposite approach for each kind of classifiers.

	FFNN											
	Basic						Trainable					
	Accuracy	Precision	Recall	TNR	F1	ROCAUC	Accuracy	Precision	Recall	TNR	F1	ROCAUC
02/14	0.9997	0.9956	0.9979	0.9998	0.9968	0.9995	**0.9999**	**1.0000**	0.9979	**1.0000**	**0.9989**	**0.9996**
02/15	**0.9797**	**0.9832**	0.9228	**0.9955**	**0.9507**	**0.9984**	0.9153	0.8613	**0.9403**	0.9083	0.8721	0.9929
02/16	0.9989	0.9669	**1.0000**	0.9989	0.9825	1.0000	**1.0000**	**0.9998**	0.9999	**1.0000**	**0.9999**	1.0000
02/22	0.9947	0.7517	**0.9656**	0.9952	0.8420	0.9800	**0.9981**	**0.9897**	0.8734	**0.9999**	**0.9248**	**0.9830**
02/23	**0.9961**	0.8252	**0.8573**	0.9978	**0.8365**	**0.9984**	0.9914	**0.9118**	0.3467	**0.9996**	0.4980	0.9274
03/02	**0.8630**	0.8980	**0.7666**	0.9293	**0.8180**	**0.9566**	0.8620	**0.9401**	0.7071	**0.9686**	0.8052	0.9363
	XGBoost											
	Basic						Trainable					
	Accuracy	Precision	Recall	TNR	F1	ROCAUC	Accuracy	Precision	Recall	TNR	F1	ROCAUC
02/14	0.9999	0.9998	0.9979	1.0000	0.9988	0.9989	0.9999	0.9998	0.9979	1.0000	0.9988	0.9989
02/15	0.9968	0.9856	1.0000	0.9959	0.9927	**0.9998**	**0.9969**	**0.9860**	1.0000	**0.9961**	**0.9929**	0.9980
02/16	1.0000	0.9998	1.0000	1.0000	0.9999	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000
02/22	0.9985	0.9385	**0.9531**	0.9991	0.9457	**0.9987**	**0.9991**	**1.0000**	0.9375	**1.0000**	**0.9677**	0.9909
02/23	0.9988	0.9146	1.0000	0.9988	0.9554	1.0000	**1.0000**	**1.0000**	1.0000	**1.0000**	**1.0000**	1.0000
03/02	0.9883	0.9726	**0.9995**	0.9806	0.9858	**0.9995**	**0.9913**	**0.9816**	0.9974	**0.9871**	**0.9894**	0.9966
	KitsuneAD											
	Basic						Trainable					
	Accuracy	Precision	Recall	TNR	F1	ROCAUC	Accuracy	Precision	Recall	TNR	F1	ROCAUC
02/14	0.9504	0.4748	0.2980	0.9834	0.3662	0.9608	**0.9999**	**0.9996**	**0.9979**	**1.0000**	**0.9987**	**0.9988**
02/15	0.9115	0.7106	**1.0000**	0.8870	0.8308	0.9945	**0.9954**	**0.9888**	0.9899	**0.9969**	**0.9893**	**0.9995**
02/16	0.9154	**0.2523**	**1.0000**	0.9130	**0.4030**	**0.9831**	**0.9715**	0.0000	0.0000	**1.0000**	0.0000	0.5000
02/22	0.7123	0.0449	**0.9688**	0.7087	0.0859	**0.9204**	**0.9880**	**0.8000**	0.1875	**0.9993**	**0.3038**	0.5914
02/23	0.9268	0.0744	0.4267	0.9331	0.1267	0.9109	**0.9977**	**0.9841**	**0.8267**	**0.9998**	**0.8986**	**0.9764**
03/02	0.8350	0.8883	0.6810	**0.9411**	0.7709	0.8694	**0.9455**	**0.9412**	0.9240	**0.9602**	**0.9325**	**0.9773**

Table 5.5: Results of session-level classification for other metrics: Accuracy, Precision, Recall, TNR, F1-measure, and ROCAUC. Basic column shows results of the Basic approach and Trainable column shows results of the Trainable approach. The bold values mean that the values are better than that achieved by the opposite approach for each kind of classifiers.

		FFNN			
		Basic		Trainable	
	n_samples	n_detected	Recall	n_detected	Recall
SSH-BruteForce(02/14)	4715	4705.0	0.9979	4705.0	0.9979
DoS-Slowloris(02/15)	2326	1785.4	0.7676	**2057.8**	**0.8847**
DoS-GoldenEye(02/15)	4705	**4702.9**	**0.9996**	4553.7	0.9678
DoS-SlowHTTPTest(02/16)	4705	**4705.0**	**1.0000**	4704.7	0.9999
BruteForce-XSS(02/22)	14	**14.0**	**1.0000**	13.4	0.9571
BruteForce-Web(02/22)	45	**45.0**	**1.0000**	42.5	0.9444
SQL Injection(02/22)	5	**2.8**	**0.5600**	0.0	0.0000
SQL Injection(02/23)	11	**5.2**	**0.4727**	0.1	0.0091
BruteForce-XSS(02/23)	24	**24.0**	**1.0000**	23.7	0.9875
BruteForce-Web(02/23)	40	**35.1**	**0.8775**	2.2	0.0550
Bot(03/02)	30561	**23429.2**	**0.7666**	21609.1	0.7071
		XGBoost			
		Basic		Trainable	
	n_samples	n_detected	Recall	n_detected	Recall
SSH-BruteForce(02/14)	4715	4705	0.9979	4705	0.9979
DoS-Slowloris(02/15)	2326	2326	1.0000	2326	1.0000
DoS-GoldenEye(02/15)	4705	4705	1.0000	4705	1.0000
DoS-SlowHTTPTest(02/16)	4705	4705	1.0000	4705	1.0000
BruteForce-XSS(02/22)	14	14	1.0000	14	1.0000
BruteForce-Web(02/22)	45	45	1.0000	45	1.0000
SQL Injection(02/22)	5	**2**	**0.4000**	1	0.2000
SQL Injection(02/23)	11	11	1.0000	11	1.0000
BruteForce-XSS(02/23)	24	24	1.0000	24	1.0000
BruteForce-Web(02/23)	40	40	1.0000	40	1.0000
Bot(03/02)	30561	**30545**	**0.9995**	30482	0.9974
		KitsuneAD			
		Basic		Trainable	
	n_samples	n_detected	Recall	n_detected	Recall
SSH-BruteForce(02/14)	4715	1405	0.2980	**4705**	**0.9979**
DoS-Slowloris(02/15)	2326	**2326**	**1.0000**	2320	0.9974
DoS-GoldenEye(02/15)	4705	4705	1.0000	**4703**	**0.9996**
DoS-SlowHTTPTest(02/16)	4705	**4705**	**1.0000**	0	0.0000
BruteForce-XSS(02/22)	14	**14**	**1.0000**	12	0.8571
BruteForce-Web(02/22)	45	**45**	**1.0000**	0	0.0000
SQL Injection(02/22)	5	**3**	**0.6000**	0	0.0000
SQL Injection(02/23)	11	0	0.0000	0	0.0000
BruteForce-XSS(02/23)	24	24	1.0000	24	1.0000
BruteForce-Web(02/23)	40	8	0.2000	**38**	**0.9500**
Bot(03/02)	30561	20811	0.6810	**29155**	**0.9540**

Table 5.6: Numbers of sessions in each class and the class-wise results of session-level classification . The bold values mean that the values are better than that achieved by the opposite approach for each kind of classifiers.

	TP	FP	TN	FN	Accuracy	Precision	Recall	TNR	F1	n_processed
$k = 1$	46948	**756**	**333017**	203	0.9975	**0.9842**	0.9957	**0.9977**	0.9899	1
$k = 5$	47107	891	332882	44	0.9975	0.9814	0.9991	0.9973	**0.9902**	4.1344
$k = 10$	47120	960	332813	31	0.9974	0.9800	0.9993	0.9971	0.9896	7.3950
$k = 15$	47121	964	332809	30	0.9974	0.9800	0.9994	0.9971	0.9896	9.1985
$k = \infty$	**47122**	977	332796	**29**	0.9974	0.9797	0.9994	0.9971	0.9894	16.9571

Table 5.7: Results of the simulation of the first k prediction procedure with XGBoost classifiers. The bold values are the best values of each metric among all k.

	$k = 1$		$k = 5$		$k = 10$		$k = 15$		$k = \infty$	
	Recall	n_processed	Recall	n_processed	Recall	n_processed	Recall	n_processed	Recall	n_processed
SSH-BruteForce(02/14)	0.9972	1	0.9975	1.0045	0.9979	1.0053	0.9979	1.0053	0.9979	1.0053
DoS-GoldenEye(02/15)	0.9996	1	0.9998	1.0011	1.0000	1.0015	1.0000	1.0015	1.0000	1.0015
DoS-Slowloris(02/15)	0.9991	1	1.0000	1.0021	1.0000	1.0021	1.0000	1.0021	1.0000	1.0021
DoS-SlowHTTPTest(02/16)	0.9989	1	1.0000	1.0021	1.0000	1.0021	1.0000	1.0021	1.0000	1.0021
BruteForce-Web(02/22)	0.9556	1	0.9778	1.1778	1.0000	1.2667	1.0000	1.2667	1.0000	1.2667
BruteForce-XSS(02/22)	0.9286	1	0.9286	1.2857	0.9286	1.6429	1.0000	1.7857	1.0000	1.7857
SQL Injection(02/22)	0.0000	1	0.0000	5.0000	0.4000	8.4000	0.4000	8.6000	0.4000	8.6000
BruteForce-Web(02/23)	0.9750	1	0.9750	1.1000	0.9750	1.2250	0.9750	1.3500	1.0000	1.4250
BruteForce-XSS(02/23)	1.0000	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
SQL Injection(02/23)	1.0000	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Bot(03/02)	0.9944	1	0.9992	1.0085	0.9995	1.0115	0.9995	1.0115	0.9995	1.0115

Table 5.8: Class-wise Recalls and averages of the number of processed packets using first k prediction with XGBoost classifiers

Table 5.9: Number of samples before splitting

	n_packets	n_sessions	n_sessions / n_packets
BENIGN(02/14)	3667015	280354	0.076
SSH-BruteForce(02/14)	4237098	14146	0.003
BENIGN(02/15)	875580	76065	0.087
DoS-GoldenEye(02/15)	258137	14116	0.055
DoS-Slowloris(02/15)	100237	6979	0.070
BENIGN(02/16)	6366534	480943	0.076
DoS-SlowHTTPTest(02/16)	211100	14116	0.067
BENIGN(02/22)	158330	13578	0.086
BruteForce-Web(02/22)	18252	137	0.008
BruteForce-XSS(02/22)	11699	42	0.004
SQL Injection(02/22)	178	17	0.096
BENIGN(02/23)	189897	17851	0.094
BruteForce-Web(02/23)	16536	123	0.007
BruteForce-XSS(02/23)	22342	72	0.003
SQL Injection(02/23)	349	31	0.089
BENIGN(03/02)	2321495	133252	0.057
Bot(03/02)	985375	91685	0.093
Total	19440154	1143507	0.059

		FFNN			
		Basic		Trainable	
	packet-level	session-level	session-level / packet-level	session-level	session-level / packet-level
SSH-BruteForce(02/14)	1374640.2	4705.0	0.0034	4705.0	0.0034
DoS-GoldenEye(02/15)	73801.9	4702.9	0.0663	4524.1	0.0637
DoS-Slowloris(02/15)	22964.2	1785.4	0.0789	2290.3	0.1085
DoS-SlowHTTPTest(02/16)	70240.7	4705.0	0.0670	4704.4	0.0670
BruteForce-Web(02/22)	5991.4	45.0	0.0075	42.7	0.0071
BruteForce-XSS(02/22)	3504.1	14.0	0.0040	13.3	0.0038
SQL Injection(02/22)	10.1	2.8	0.3016	0.1	0.0167
BruteForce-Web(02/23)	1167.7	36.0	0.1193	25.3	0.0675
BruteForce-XSS(02/23)	5298.5	24.0	0.0068	24.0	0.0068
SQL Injection(02/23)	11.7	4.3	0.4347	0.2	0.0108
Bot(03/02)	220767.0	23429.2	0.1059	23782.2	0.1091
Total	1778397.5	39453.6	0.0222	40111.6	0.0226
		XGBoost			
		Basic		Trainable	
	packet-level	session-level	session-level / packet-level	session-level	session-level / packet-level
SSH-BruteForce(02/14)	1407076	4705	0.0033	4705	0.0033
DoS-GoldenEye(02/15)	85830	4705	0.0548	4705	0.0548
DoS-Slowloris(02/15)	33188	2326	0.0701	2326	0.0701
DoS-SlowHTTPTest(02/16)	70352	4705	0.0669	4705	0.0669
BruteForce-Web(02/22)	6060	45	0.0074	45	0.0074
BruteForce-XSS(02/22)	3579	14	0.0039	14	0.0039
SQL Injection(02/22)	9	2	0.2222	1	0.1111
BruteForce-Web(02/23)	5428	40	0.0074	40	0.0074
BruteForce-XSS(02/23)	7416	24	0.0032	24	0.0032
SQL Injection(02/23)	117	11	0.0940	11	0.0940
Bot(03/02)	326892	30545	0.0934	30482	0.0932
Total	1945947	47122	0.0242	47058	0.0242
		KitsuneAD			
		Basic		Trainable	
	packet-level	session-level	session-level / packet-level	session-level	session-level / packet-level
SSH-BruteForce(02/14)	39601	1405	0.0355	4705	0.1188
DoS-GoldenEye(02/15)	85995	4705	0.0547	4703	0.0547
DoS-Slowloris(02/15)	33243	2326	0.0700	2320	0.0698
DoS-SlowHTTPTest(02/16)	69940	4705	0.0673	0	0.0000
BruteForce-Web(02/22)	6067	45	0.0074	0	0.0000
BruteForce-XSS(02/22)	3592	14	0.0039	12	0.0033
SQL Injection(02/22)	21	3	0.1429	0	0.0000
BruteForce-Web(02/23)	10	8	0.8000	38	3.8000
BruteForce-XSS(02/23)	1943	24	0.0124	24	0.0124
SQL Injection(02/23)	0	0	0.0000	0	0.0000
Bot(03/02)	180570	20811	0.1153	29155	0.1615
Total	420982	34046	0.0809	40957	0.0973

Table 5.10: Number of detected packets and sessions

Approach	Accuracy	Precision	Recall	F1
Our XGBoost($k = \infty$)	0.9974	0.9797	0.9994	0.9894
Our XGBoost($k = 1$)	0.9975	0.9842	0.9957	0.9899
Our XGBoost ($k = 5$)	0.9975	0.9814	0.9991	0.9902
Atefinia and Ahmad [75]	1.0000	1.0000	1.0000	1.0000
Basnet et al. [79]	0.9900	NA	NA	NA
Catillo et al. [80]	0.9920	0.9500	0.9890	0.9691
Chadza et al. [81]	0.9700	NA	NA	NA
D'hooge et al. [82]	0.9600	0.9900	0.7900	0.8788
Ferrag et al. [33]	0.9738	NA	0.9818	NA
Filho et al. [76]	NA	1.0000	1.0000	1.0000
Fitni and Ramli [83]	0.9880	0.9880	0.9710	0.9794
Gamage and Samarabandu [84]	0.9840	0.9779	0.9827	0.9803
Hua [85]	0.9837	0.9814	0.9837	0.9825
Huancayo Ramos et al. [86]	0.9999	1.0000	0.9999	0.9999
Kanimozhi and Jacob [77]	1.0000	1.0000	1.0000	1.0000
Kanimozhi and Jacob [78]	0.9997	0.9996	1.0000	0.9998
Karatas et al. [87]	0.9969	0.9970	0.9969	0.9969
Kim et al. [88]	0.9999	0.8175	0.8225	0.8200
Li et al. [89]	NA	NA	1.0000	NA
Lin et al. [90]	0.9620	0.9600	0.9600	0.9600
Zhao et al. [91]	0.9790	0.9800	0.9800	0.9800

Table 5.11: Comparison to results in references. Accuracies, Precisions and Recalls of the references were cited from [74] and F1-measures were calculated from them.

Chapter 6

Conclusion

Finally we summarize the contributions of this research on both the MDL estimators and the machine learninge based cyber security.

In this thesis, we have developed the improved MDL estimators based on fiber bundles of local exponential families such that good performance guarantees hold for non-exponential families. As a result, we obtained the MDL estimators for non-exponential families with the similar performance as for exponential families.

We also have discussed its applications to the mixture families and the contaminated Gaussian location families. For the mixture families, we showed that the mixture families satisfy the requirements for our improved estimators and therefore we can apply the results for general cases to the mixture families. For the contaminated Gaussian location families, we saw that these models do not satisfy the requirements in general. Therefore we expanded our performance guarantee to the conditional form and introduced the notion of typical set of data strings. As a result, we showed that the MDL estimators for the contaminated Gaussian location families typically achieve good performance.

We also have seen that the requirement of parameter restriction for Gaussian distributions in the MDL context through the evaluation of the Fisher information on the specific parametrization. In concrete, we employed eigen values of covariance matrix as parameters for Gaussian distribution implicitly and evaluated the Fisher information on such parametriza-

tion. As a result, we saw that the restriction of region of the eigen values was needed to define the factor of the Jeffreys' prior distribution.

In this thesis, we also have proposed a method of network intrusion detection based on packet-level classification using machine learning techniques.

In the proposed method, we conduct packet-level classification. Then we generate flow-level results based on the packet-level results. We also have seen the performance of the proposed method by conducting experiments using a benchmark dataset. The experimental results showed that the proposed method could achieve high performance in both packet-level and flow-level. The experimental results also showed that the proposed method might have an advantage in terms of the quickness of detection.

Through this thesis, we have discussed both theoretical and practical topics on machine learning. The results will help future researches and developments in machine learning and cyber security.

Acknowledgments

First of all, I would like to thank my supervisor, Prof. Jun'ichi Takeuchi. Without his continuous support including encouraging me and technical advice, I could not finish this research.

I would like to thank Dr. Masanori Kawakita from Mie Toyopet Corporation, who was an assistant professor of Kyushu university. Some results on MDL estimators in this research is thanks to his advice. I also would like to thank Prof. Andrew R. Barron from Yale University. He is one of originators of an important theorem, which this research is based on, and gave me interesting comments on this research. I also would like to thank Dr. Chansu Han, Dr. Tao Ban and Dr. Takeshi Takahashi from National Institute of Information and Communications Technology. They helped me in the research on cyber security with their expertise on it.

I also would like to members of the advisory committee for this research and the examiners of this thesis. Prof. Eiji Takimoto from Kyushu University is a member of advisory committee for this research and the chief examiner of this thesis. Associate Prof. Kazuho Watanabe from Toyohashi University of technology is a member of advisory committee for this research. Prof. Kohei Hatano from Kyushu University is a sub examiner of this thesis. They kindly gave me advise on the research and on writing this thesis.

Finally, I would like to thank my parents. Without their help, I could not complete this research and student life.

References

[1] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[2] A. R. Barron and T. M. Cover, "Minimum complexity density estimation," *IEEE Trans. on Inf. Theory*, vol. 37, no. 4, pp. 1034–1054, 1991.

[3] S. Chatterjee and A. R. Barron, "Information theory of penalized likelihoods and its statistical implications," *arXiv:1401.6714v2*, pp. 1–17, 2014.

[4] A. Rényi, "On measures of entropy and information," in *Proc. of the Fourth Berkeley Symp. on Math. Stat. and Prob.*, vol. 1, 1961, pp. 547–561.

[5] T. van Erven and P. Harremoës, "Rényi divergence and Kullback-Leibler divergence," *IEEE Trans. on Inf. Theory*, vol. 60, no. 7, pp. 3797–3820, 2014.

[6] A. R. Barron, "Logically smooth density estimation," Ph.D. dissertation, Stanford University, 1985.

[7] P. D. Grünwald, *The minimum description length principle*. MIT press, 2007.

[8] Y. M. Shtar'kov, "Universal sequential coding of single messages," *Problemy Peredachi Informatsii*, vol. 23, no. 3, pp. 3–17, 1987.

[9] J. Rissanen, "Fisher information and stochastic complexity," *IEEE Trans. on Inf. Theory*, vol. 42, no. 1, pp. 40–47, 1996.

[10] J. Takeuchi and A. R. Barron, "Asymptotically minimax regret by Bayes mixtures," in *Proc. IEEE International Symp. on Inf. Theory*, 1998, p. 318.

[11] S. Amari and H. Nagaoka, *Methods of information geometry*. American Mathematical Soc., 2007, vol. 191.

[12] J. Takeuchi and A. R. Barron, "Asymptotically minimax regret by Bayes mixtures for non-exponential families," in *2013 IEEE Information Theory Workshop (ITW)*, 2013, pp. 1–5.

[13] ——, "Asymptotically minimax regret for models with hidden variables," in *Proc. IEEE International Symp. on Inf. Theory*, 2014, pp. 3037–3041.

[14] ——, "Stochastic complexity for tree models," in *2014 IEEE Information Theory Workshop (ITW 2014)*, 2014, pp. 222–226.

[15] Q. J. Li, "Estimation of mixture models," Ph.D. dissertation, Yale University, 1999.

[16] T. Zhang, "On the convergence of mdl density estimation," in *International Conference on Computational Learning Theory*. Springer, 2004, pp. 315–330.

[17] A. R. Barron, C. Huang, J. Q. Li, and X. Luo, "Mdl, penalized likelihood, and statistical risk," in *2008 IEEE Information Theory Workshop*, 2008, pp. 247–257.

[18] T. Zhang, "From ϵ-entropy to kl-entropy: Analysis of minimum information complexity density estimation," vol. 34, no. 5, 2006, pp. 2180–2210.

[19] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[20] ——, *Robust Statistics*. Wiley & Sons, Hoboken, NJ, USA, 2004.

[21] J. W. Tukey, "A survey of sampling from contaminated distributions. in: Oklin, i., ed., contributions to probability and statistics," 1960.

[22] M. Kawakita and J. Takeuchi, "Barron and Cover's theory in supervised learning and its application to lasso," in *Proc. of the 33rd International Conference on Machine Learning*, 2016, pp. 1958–1966.

[23] ——, "Minimum description length principle in supervised learning with application to lasso," accepted for publication, *IEEE Trans. on Inf. Theory*, May 2020.

[24] E. U. A. for Cybersecurity, "Enisa threat landscape 2021," https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021, 2021, [Online; accessed 30th January 2022].

[25] ——, "Enisa threat landscape 2022," https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022, 2022, [Online; accessed 30th January 2023].

[26] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 686–728, 2019. [Online]. Available: https://doi.org/10.1109/comst.2018.2847722

[27] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019. [Online]. Available: https://doi.org/10.1109/access.2019.2895334

[28] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, 2018. [Online]. Available: https://doi.org/10.14722/ndss.2018.23204

[29] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan, and R. Agrawal, "Intrusion detection using payload embeddings," *IEEE Access*, vol. 10, pp. 4015–4030, 2022. [Online]. Available: https://doi.org/10.1109/access.2021.3139835

[30] Y. Yu, J. Long, and Z. Cai, "Session-based network intrusion detection using a deep learning architecture," in *Modeling Decisions for Artificial Intelligence*.

Springer International Publishing, 2017, pp. 144–155. [Online]. Available: https://doi.org/10.1007/978-3-319-67422-3_13

[31] C. Chen, X. Xu, G. Wang, and L. Yang, "Network intrusion detection model based on neural network feature extraction and PSO-SVM," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*. IEEE, Apr. 2022. [Online]. Available: https://doi.org/10.1109/icsp54964.2022.9778404

[32] B. Caulkins, J. Lee, and M. Wang, "Packet- vs. session-based modeling for intrusion detection systems," in *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*. IEEE, 2005. [Online]. Available: https://doi.org/10.1109/itcc.2005.222

[33] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020. [Online]. Available: https://doi.org/10.1016/j.jisa.2019.102419

[34] H. Hindy, D. Brosset, E. Bayne, A. K. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104 650–104 675, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.3000179

[35] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010. [Online]. Available: https://doi.org/10.1109/sp.2010.25

[36] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. [Online]. Available: https://doi.org/10.1109/comst.2015.2494502

[37] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.06.005

[38] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*. IEEE Comput. Soc, 2000. [Online]. Available: https://doi.org/10.1109/discex.2000.821506

[39] 1998 DARPA Intrusion Detection Evaluation Dataset, 1998, Accessed: Mar. 15, 2022. [Online]. Available: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset

[40] KDD Cup 1999 Data, 1999, Accessed: Mar. 15, 2022. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[41] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, Jul. 2009. [Online]. Available: https://doi.org/10.1109/cisda.2009.5356528

[42] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*. IEEE, Nov. 2015. [Online]. Available: https://doi.org/10.1109/milcis.2015.7348942

[43] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, May 2012. [Online]. Available: https://doi.org/10.1016/j.cose.2011.12.012

[44] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for NIDS evaluation," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. ACM, Apr. 2011. [Online]. Available: https://doi.org/10.1145/1978672.1978676

[45] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, Sep. 2014. [Online]. Available: https://doi.org/10.1016/j.cose.2014.05.011

[46] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 2018. [Online]. Available: https://doi.org/10.5220/0006639801080116

[47] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Applied Sciences*, vol. 9, no. 16, p. 3414, Aug. 2019. [Online]. Available: https://doi.org/10.3390/app9163414

[48] H. Gwon, C. Lee, R. Keum, and H. Choi, "Network intrusion detection based on lstm and feature embedding," 2019. [Online]. Available: https://arxiv.org/abs/1911.11552

[49] R. Shrestha, A. Omidkar, S. A. Roudi, R. Abbas, and S. Kim, "Machine-learning-enabled intrusion detection system for cellular connected UAV networks," *Electronics*, vol. 10, no. 13, p. 1549, Jun. 2021. [Online]. Available: https://doi.org/10.3390/electronics10131549

[50] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, and F. D. Turck, "Towards model generalization for intrusion detection: Unsupervised machine learning techniques,"

Journal of Network and Systems Management, vol. 30, no. 1, Oct. 2021. [Online]. Available: https://doi.org/10.1007/s10922-021-09615-7

[51] K. Park, Y. Song, and Y.-G. Cheong, "Classification of attack types for intrusion detection systems using a machine learning algorithm," in *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, Mar. 2018. [Online]. Available: https://doi.org/10.1109/bigdataservice.2018.00050

[52] W. Fang, X. Tan, and D. Wilbur, "Application of intrusion detection technology in network safety based on machine learning," *Safety Science*, vol. 124, p. 104604, Apr. 2020. [Online]. Available: https://doi.org/10.1016/j.ssci.2020.104604

[53] E. L. Goodman, C. Zimmerman, and C. Hudson, "Packet2vec: Utilizing word2vec for feature extraction in packet data," 2020. [Online]. Available: https://arxiv.org/abs/2004.14477

[54] D. Gaikwad and R. C. Thool, "Intrusion detection system using bagging ensemble method of machine learning," in *2015 International Conference on Computing Communication Control and Automation*. IEEE, Feb. 2015. [Online]. Available: https://doi.org/10.1109/iccubea.2015.61

[55] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2016. [Online]. Available: https://doi.org/10.1109/etfa.2016.7733515

[56] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, and F. D. Turck, "Unsupervised machine learning techniques for network intrusion detection on modern data," in *2020 4th Cyber Security in Networking Conference (CSNet)*. IEEE, Oct. 2020. [Online]. Available: https://doi.org/10.1109/csnet50428.2020.9265461

[57] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow datasets for machine learning-based network intrusion detection systems," in *Lecture Notes of*

the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer International Publishing, 2021, pp. 117–135. [Online]. Available: https://doi.org/10.1007/978-3-030-72802-1_9

[58] A. Corsini, S. J. Yang, and G. Apruzzese, "On the evaluation of sequential machine learning for network intrusion detection," in *The 16th International Conference on Availability, Reliability and Security*. ACM, Aug. 2021. [Online]. Available: https://doi.org/10.1145/3465481.3470065

[59] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the CICIDS2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, May 2021. [Online]. Available: https://doi.org/10.1109/spw53761.2021.00009

[60] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE, Sep. 2017. [Online]. Available: https://doi.org/10.1109/sisy.2017.8080566

[61] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset)," in *Selected Papers from the 12th International Networking Conference*. Springer International Publishing, 2021, pp. 73–84. [Online]. Available: https://doi.org/10.1007/978-3-030-64758-2_6

[62] K. Masumi, C. Han, T. Ban, and T. Takahashi, "Towards efficient labeling of network incident datasets using tcpreplay and snort," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*. ACM, Apr. 2021. [Online]. Available: https://doi.org/10.1145/3422337.3450326

[63] R. Ishibashi, H. Goto, C. Han, T. Ban, T. Takahashi, and J. Takeuchi, "Which packet did they catch? associating NIDS alerts with their communication sessions," in *2021 16th*

Asia Joint Conference on Information Security (AsiaJCIS). IEEE, Aug. 2021. [Online]. Available: https://doi.org/10.1109/asiajcis53848.2021.00012

[64] R. Ishibashi, K. Miyamoto, C. Han, T. Ban, T. Takahashi, and J. Takeuchi, "Generating labeled training datasets towards unified network intrusion detection systems," *IEEE Access*, vol. 10, pp. 53 972–53 986, 2022. [Online]. Available: https://doi.org/10.1109/access.2022.3176098

[65] T. Takahashi, Y. Umemura, C. Han, T. Ban, K. Furumoto, O. Nakamura, K. Yoshioka, J. Takeuchi, N. Murata, and Y. Shiraishi, "Designing comprehensive cyber threat analysis platform: Can we orchestrate analysis engines?" in *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, Mar. 2021. [Online]. Available: https://doi.org/10.1109/percomworkshops51409.2021.9431125

[66] K. Miyamoto, H. Goto, R. Ishibashi, C. Han, T. Ban, T. Takahashi, and J. Takeuchi, "Malicious packet classification based on neural network using kitsune features," in *Intelligent Systems and Pattern Recognition*. Springer International Publishing, 2022, pp. 306–314. [Online]. Available: https://doi.org/10.1007/978-3-031-08277-1_25

[67] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[68] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002. [Online]. Available: https://doi.org/10.1613/jair.953

[69] "SMOTE - Version 0.10.1," https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html, [Online; accessed 16th February 2023.

[70] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining. ACM, Aug. 2016. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[71] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, Oct. 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451

[72] ymirsky, "Kitsune-py," https://github.com/ymirsky/Kitsune-py (last visited on 2021-12-31).

[73] M. Iida, K. Miyamoto, S. Kawanaka, Y. Takeishi, C. Han, T. Ban, T. Takahashi, and J. Takeuchi, "Multimodal feature integration for high-accuracy network intrusion detection," in *6th International Conference on Computer Applications and Information Security (ICCAIS)*, Mar. 2023.

[74] J. L. Leevy and T. M. Khoshgoftaar, "A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 big data," *Journal of Big Data*, vol. 7, no. 1, Nov. 2020. [Online]. Available: https://doi.org/10.1186/s40537-020-00382-x

[75] R. Atefinia and M. Ahmadi, "Network intrusion detection using multi-architectural modular deep neural network," *The Journal of Supercomputing*, vol. 77, no. 4, pp. 3571–3593, Aug. 2020. [Online]. Available: https://doi.org/10.1007/s11227-020-03410-y

[76] F. S. de Lima Filho, F. A. F. Silveira, A. de Medeiros Brito Junior, G. Vargas-Solar, and L. F. Silveira, "Smart detection: An online approach for DoS/DDoS attack detection using machine learning," *Security and Communication Networks*, vol. 2019, pp. 1–15, Oct. 2019. [Online]. Available: https://doi.org/10.1155/2019/1574749

[77] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," *ICT Express*, vol. 5, no. 3, pp. 211–214, Sep. 2019. [Online]. Available: https://doi.org/10.1016/j.icte.2019.03.003

[78] V. Kanimozhi and D. T. P. Jacob, "CALIBRATION OF VARIOUS OPTIMIZED MACHINE LEARNING CLASSIFIERS IN NETWORK INTRUSION DETECTION SYSTEM ON THE REALISTIC CYBER DATASET CSE-CIC-IDS2018 USING CLOUD COMPUTING," *International Journal of Engineering Applied Sciences and Technology*, vol. 04, no. 06, pp. 209–213, Dec. 2019. [Online]. Available: https://doi.org/10.33564/ijeast.2019.v04i06.036

[79] R. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, "Towards detecting and classifying network intrusion traffic using deep learning frameworks," 12 2019.

[80] M. Catillo, M. Rak, and U. Villano, "2l-ZED-IDS: A two-level anomaly detector for multiple attack classes," in *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2020, pp. 687–696. [Online]. Available: https://doi.org/10.1007/978-3-030-44038-1_63

[81] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan, "Contemporary sequential network attacks prediction using hidden markov model," in *2019 17th International Conference on Privacy, Security and Trust (PST)*. IEEE, Aug. 2019. [Online]. Available: https://doi.org/10.1109/pst47121.2019.8949035

[82] L. D'hooge, T. Wauters, B. Volckaert, and F. D. Turck, "Inter-dataset generalization strength of supervised machine learning methods for intrusion detection," *Journal of Information Security and Applications*, vol. 54, p. 102564, Oct. 2020. [Online]. Available: https://doi.org/10.1016/j.jisa.2020.102564

[83] Q. R. S. Fitni and K. Ramli, "Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems," in *2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*. IEEE, Jul. 2020. [Online]. Available: https://doi.org/10.1109/iaict50021.2020.9172014

[84] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion

detection: A survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, p. 102767, Nov. 2020. [Online]. Available: https://doi.org/10.1016/j.jnca.2020.102767

[85] Y. Hua, "An efficient traffic classification scheme using embedded feature selection and LightGBM," in *2020 Information Communication Technologies Conference (ICTC)*. IEEE, May 2020. [Online]. Available: https://doi.org/10.1109/ictc49638.2020.9123302

[86] K. S. H. Ramos, M. A. S. Monge, and J. M. Vidal, "Benchmark-based reference model for evaluating botnet detection tools driven by traffic-flow analytics," *Sensors*, vol. 20, no. 16, p. 4501, Aug. 2020. [Online]. Available: https://doi.org/10.3390/s20164501

[87] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32 150–32 162, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.2973219

[88] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, Jun. 2020. [Online]. Available: https://doi.org/10.3390/electronics9060916

[89] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Computers & Security*, vol. 95, p. 101851, Aug. 2020. [Online]. Available: https://doi.org/10.1016/j.cose.2020.101851

[90] P. Lin, K. Ye, and C.-Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Cloud Computing – CLOUD 2019*. Springer International Publishing, 2019, pp. 161–176. [Online]. Available: https://doi.org/10.1007/978-3-030-23502-4_12

[91] F. Zhao, H. Zhang, J. Peng, X. Zhuang, and S.-G. Na, "A semi-self-taught network intrusion detection system," *Neural Computing and Applications*, vol. 32, no. 23, pp. 17 169–17 179, Apr. 2020. [Online]. Available: https://doi.org/10.1007/s00521-020-04914-7