Position: C^*-Algebraic Machine Learning — Moving in a New Direction

Yuka Hashimoto [1 2] **Masahiro Ikeda** [2 3] **Hachem Kadri** [4]

Abstract

Machine learning has a long collaborative tradition with several fields of mathematics, such as statistics, probability and linear algebra. We propose a new direction for machine learning research: *C^*-algebraic ML*—a cross-fertilization between C^*-algebra and machine learning. The mathematical concept of C^*-algebra is a natural generalization of the space of complex numbers. It enables us to unify existing learning strategies, and construct a new framework for more diverse and information-rich data models. We explain why and how to use C^*-algebras in machine learning, and provide technical considerations that go into the design of C^*-algebraic learning models in the contexts of kernel methods and neural networks. Furthermore, we discuss open questions and challenges in C^*-algebraic ML and give our thoughts for future development and applications.

1. Introduction

Machine learning problems and methods are currently becoming more and more complicated. We have many types of structured data, such as time-series data, image data, and graph data. In addition, not only are the models large, but multiple models and tasks have to be considered in some situations.

To address these situations, we propose *C^*-algebraic machine learning*: application of C^*-algebra to machine learning methods. Typical examples of C^*-algebras are the space of continuous functions on a compact space and the space of bounded linear operators on a Hilbert space. C^*-algebra was first proposed in quantum mechanics to model physical observables and has been investigated in pure mathematics, mathematical physics, and quantum mechanics. Whereas its rich mathematical and theoretical investigations, its main application is limited to quantum mechanics. In the current situation in machine learning, we believe that it is time to apply these rich investigations to machine learning methods. Since C^*-algebras enable us to unify complex values, matrices, functions, and linear operators, we expect that the generalization of machine learning methods using C^*-algebras allows us to unify existing methods and construct a framework for more complicated data and models. Figure 1 shows an overview of the C^*-algebraic machine learning.

In this paper, we mainly focus on two approaches: kernel methods and neural networks. For kernel methods, most of existing methods are realized using reproducing kernel Hilbert spaces (RKHSs) or vector-valued RKHS (vvRKHS), which are constructed by positive definite kernels (Schölkopf & Smola, 2001; Saitoh & Sawano, 2016). The reproducing property enables us to evaluate the value of a function at a point using the inner product, which makes it easy for us to implement algorithms and analyze them theoretically. Moreover, we can apply kernel methods to probabilistic and statistical settings by embedding probability measures in an RKHS. This embedding is called the kernel mean embedding. However, since RKHSs (resp. vvRKHSs) are complex- (resp. vector-) valued function spaces, the output of the models is usually complex- or vector-valued. In addition, appropriate ways of the construction of positive definite kernels are not trivial. The generalization of RKHS by means of the C^*-algebra enables us to output more general data, such as functions and operators (Hashimoto et al., 2021). Moreover, C^*-algebras give us a method to construct C^*-algebra-valued positive definite kernels for structured data (Hasimoto et al., 2023a). The noncommutative product structure in C^*-algebras ($ab \neq ba$ for elements a, b in the C^*-algebra) enables us to construct an operation that goes beyond the multiplication and convolution.

As for neural networks, the models are becoming larger and more complicated. For example, in ensemble learning (Dong et al., 2020; Ganaie et al., 2022), multitask learning (Zhang et al., 2014; Ruder et al., 2019), and meta-learning (Ravi & Larochelle, 2017; Finn et al., 2017; Rusu et al., 2019), we need to consider multiple tasks and models. In addition, large language models (LLMs) have large numbers of learning parameters and need large numbers of

[1]NTT corporation, Tokyo, Japan [2]Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan [3]Keio University, Yokohama, Japan [4]Aix-Marseille University, CNRS, LIS, Marseille, France. Correspondence to: Yuka Hashimoto <yuka.hashimoto@ntt.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

training samples. To fully extract features of data in these architectures, we expect that C^*-algebras play an important role since they enable us to represent multiple models and tasks simultaneously (Hashimoto et al., 2022). In addition, although one reason for the success of current large modes like LLMs is a large number of training samples, in some applications, we do not have enough data to train models. For example, we do not always have enough healthcare data, and for anomaly detection, we do not always have enough abnormal data. Moreover, federated learning has been investigated to analyze privacy data distributed in multiple nodes without sharing it with other nodes (McMahan et al., 2017; Bonawitz et al., 2021). In these cases, we cannot rely on the powerfulness of neural network models coming from large numbers of training samples. We believe that the rich structure of models with C^*-algebras will offer a new approach to address these situations.

In this paper, we discuss known advantages of applying C^*-algebra to machine learning, as summarized as follows:

- We can generalize the complex-valued inner product to a C^*-algebra-valued inner product. Since many machine learning methods involve the inner product, such as projection and computing correlations, the generalization can help us extract data features effectively in these methods.
- Using the C^*-algebra-valued inner product, we can generalize RKHS by means of C^*-algebra and learn function- and operator-valued maps (Subsection 5.1).
- We can design positive definite kernels for structured data using the noncommutative product (Subsection 5.1).
- We can use the norm of the C^*-algebra to alleviate the dependency of generalization error bound on the output dimension (Subsection 5.1).
- Using the generalization of kernel mean embedding by means of C^*-algebra, we can analyze operator-valued measures such as positive operator-valued measures and spectral measures (Subsection 5.1.1).
- We can continuously combine multiple models and use the tools for functions, which can be applied to ensemble, multitask, and meta-learning (Subsection 5.2).
- The noncommutative product structures in C^*-algebras induce interactions among models (Subsection 5.2.3).
- We can construct group equivariant neural networks using the products in group C^*-algebras (Subsection 5.2.3).

In each section and subsection, we discuss the above advantages in more detail. Then, we go into the technical details to show that the notion of C^*-algebra can be adapted to machine learning methods such as kernel methods and neural networks. We also provide new results showing the advantages of applying C^*-algebra. Specifically,

- By generalizing neural networks by means C^*-algebra, we can show that even if the activation functions are linear, the expressiveness of the generalized network, called C^*-algebra net, grows as the depth grows (Subsection 5.2.1).
- C^*-algebra net fills the gap between convex and nonconvex optimization for neural networks (Subsection 5.2.2).

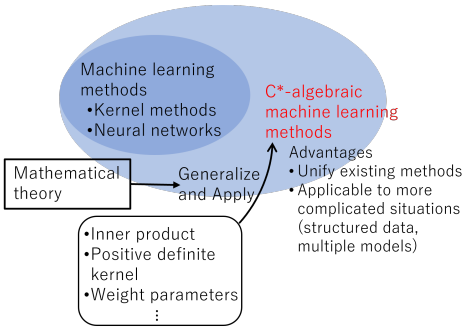

Figure 1. Overview of the C^*-algebraic machine learning

Finally, we discuss future directions of C^*-algebraic machine learning. We discuss open problems and several possible examples of C^*-algebras that could be useful in machine learning problems.

2. C^*-algebra

C^*-algebra is a natural generalization of the space of complex numbers. Thus, we can naturally generalize real- or complex-valued notions necessary to construct machine learning algorithms. In C^*-algebras, we have arithmetic operations, addition, subtraction, and multiplication, which are fundamental for arbitrary algorithms. Moreover, we have the structure of involution, which is a generalization of the complex conjugate and is necessary to generalize complex-valued notions. For example, positive definite kernels for defining RKHSs are, in general, defined as complex-valued functions. Other important notions for machine learning algorithms are the magnitude and the order. For example, to evaluate the discrepancy between two values, we consider the magnitude of the difference between the two values. In addition, we consider minimization or maximization problems in many situations. The notions of minimum and maximum are based on an order. We can generalize the order for real values to that in C^*-algebras. In the following, we will see how we can technically generalize real- or complex-valued notions to C^*-algebra-valued ones. We first recall the definition of C^*-algebra.

Definition 2.1 (C^*-algebra)**.** A set $\mathcal{A}$ is called a *C^*-algebra* if it satisfies the following three conditions:

1. $\mathcal{A}$ is an algebra over $\mathbb{C}$ and equipped with a bijection $(\cdot)^* : \mathcal{A} \to \mathcal{A}$ that satisfies for $\alpha, \beta \in \mathbb{C}$ and $c, d \in \mathcal{A}$,
   - $(\alpha c + \beta d)^* = \overline{\alpha} c^* + \overline{\beta} d^*$,
   - $(cd)^* = d^* c^*$, $\quad$ - $(c^*)^* = c$.
2. $\mathcal{A}$ is a Banach space equipped with the norm $\|\cdot\|_{\mathcal{A}}$, and for $c, d \in \mathcal{A}$, $\|cd\|_{\mathcal{A}} \leq \|c\|_{\mathcal{A}} \|d\|_{\mathcal{A}}$ holds.

3. For $c \in \mathcal{A}$, $\|c^*c\|_{\mathcal{A}} = \|c\|_{\mathcal{A}}^2$ holds. (C^*-property)

The involution $(\cdot)^*$ is a generalization of the complex conjugate. We can define an $\mathcal{A}$-valued absolute value to evaluate the magnitude of elements in $\mathcal{A}$ and compare the absolute value of elements with the partial order defined as follows.

Definition 2.2 (Positive)**.** An element $c \in \mathcal{A}$ is called *positive* if there exists $d \in \mathcal{A}$ such that $c = d^*d$. We denote $c \leq_{\mathcal{A}} d$ if $d - c$ is positive for $c, d \in \mathcal{A}$. We denote by $\mathcal{A}_+$ the subset of $\mathcal{A}$ composed of all positive elements in $\mathcal{A}$.

For $c \in \mathcal{A}$, the $\mathcal{A}$-valued absolute value $|c|_{\mathcal{A}}$ is defined as a unique element $d \in \mathcal{A}_+$ that satisfies $d^2 = c^*c$. Using the above notions, we can define an $\mathcal{A}_+$-valued objective function and optimize it in the sense of the order $\leq_{\mathcal{A}}$.

We list typical examples of C^*-algebras that are useful for machine learning below. We can find more theoretical properties and examples in, for example, Murphy (1990) and Davidson (1996).

Example 2.3.

1. Let $\mathcal{A} = C(\mathcal{Z})$, the C^*-algebra of continuous functions on a compact Hausdorff space $\mathcal{Z}$. The product of two functions $c, d \in \mathcal{A}$ is defined as $(cd)(z) = c(z)d(z)$ for $z \in \mathcal{Z}$, the involution is defined as $c^*(z) = \overline{c(z)}$, the norm is the supnorm. An element $c \in \mathcal{A}$ is positive if and only if $c(z) \geq 0$ for any $z \in \mathcal{Z}$. Note that the product is commutative, i.e., $cd = dc$ for $c, d \in \mathcal{A}$. For example, we can use $C(\mathcal{Z})$ for representing functional data and combining multiple models continuously (see Subsection 5.2).

2. Let $\mathcal{A} = \mathcal{B}(\mathcal{W})$, the C^*-algebra of bounded linear operators on a Hilbert space $\mathcal{W}$. The product is the product (the composition) of operators, the involution is the adjoint, and the norm is the operator norm. An element $c \in \mathcal{A}$ is positive if and only if c is Hermitian positive semi-definite. Note that, unlike the first example, the product is noncommutative, i.e., $cd \neq dc$ for $c, d \in \mathcal{A}$. For example, we can use $\mathcal{B}(\mathcal{W})$ for treating spectral and positive operator-valued measures (see Subsection 5.1.1).

3. If $\mathcal{W}$ is a d-dimensional space, then $\mathcal{B}(\mathcal{W})$ is the C^*-algebra of squared matrices $\mathbb{C}^{d\times d}$. The space of block diagonal squared matrix is a C^*-subalgebra of $\mathbb{C}^{d\times d}$. For example, we can use $\mathbb{C}^{d\times d}$ to represent adjoint matrices of graphs and images.

4. The group C^*-algebra on a finite discrete group G, which is denoted as $C^*(G)$, is the set of maps from G to $\mathbb{C}$. The product is defined as $(a \cdot b)(g) = \sum_{h\in G} a(h)b(h^{-1}g)$ for $g \in G$, and the adjoint is defined as $a^*(g) = \overline{a(g^{-1})}$ for $g \in G$. The norm is $\|a\| = \sup_{[\pi]\in\hat{G}} \|\pi(a)\|$, where $\hat{G}$ is the set of equivalence classes of irreducible unitary representations of G. Note that if G is an abelian group, then the product is commutative. On the other hand, if G is not an abelian group, then the product is noncommutative. For example, we can use $C^*(G)$ to construct group equivariant neural networks (see Subsection 5.2.3).

3. Representing Data and Models Using C^*-algebra

Recently, machine learning problems are getting more and more complicated. As we saw in Section 2, C^*-algebra is a natural mathematical framework to generalize the notion of complex values to functions and operators. Thus, applying functions and operators in C^*-algebras helps us deal with these complicated situations. We can effectively represent structured data and multiple models using C^*-algebras. At least, we have the following perspectives.

Data structure In many cases, data is not just composed of finite dimensional vectors but composed of time series, graphs, large images, and so on. To analyze these kinds of data with higher accuracy, we need to consider the structure of the data and represent it properly. C^*-algebra helps us represent the data structure. For example, if we have finite time-series data with a constant time interval, then we can represent the series with a finite-dimensional vector. However, if the series is infinite or if the time interval is not constant, then it is more reasonable to use a function to represent the time series. In addition, for graph data, we can use adjacent matrices to represent the graphs. Images can also be regarded as functions that map a pixel to the intensity of the pixel. Functions and matrices (operators) are perfect tools to represent the rich structure of data.

Multiple models In ensemble, multitask, and meta-learning, we consider multiple models simultaneously. In these cases, representing the models simultaneously using functions in C^*-algebras is more effective than representing each of them individually since we can use tools of functional analysis to extract common features of the models. Hashimoto et al. (2022) used integral and regression to extract common features regarding the gradients of the models.

Limited number of samples In few-shot learning, we try to train models with a limited number of samples. We often come across situations where the number of training samples is limited. For example, we do not always have enough healthcare data, biological data, abnormal data in anomaly detection, and so on. In this case, we need to extract as much information as possible from these samples. By using functions in C^*-algebras, we can represent infinitely many models, which enables us to extract a maximal amount of features.

Regarding the data structure, we can also deal with structured data such as functional data with other methods.

For example, stochastic processes (Zhu et al., 2011), operator learning (Kovachki et al., 2023), vector-valued RKHSs (Kadri et al., 2016), the framework of functional data analysis (Wang et al., 2016). Advantages of applying C^*-algebras compared to them is summarized as follows.

Product structure A C^*-algebra has the product structure. It enables us to generalize algorithms on Hilbert spaces to those on Hilbert C^*-modules. Regarding functional data, we can also use other basic function spaces such as L^2 spaces and Sobolev spaces that do not have product structures. However, the above generalizations are not possible with them. Similarly, regarding graph data, we can also vectorize a $\mathbb{C}^{d\times d}$ adjacency matrix and regard it as a d^2-dimensional vector. However, we do not have the product structure in that case. In addition, C^*-algebras allow "flexible" product structure. Depending on the C^*-algebra, we can use and take advantage of different product structures. For example, the product structure is the convolution for group C^*-algebras (See Example 2.3.4). We can also apply noncommutative product structures to induce interactions (See Subsection 5.2.3 for more details).

Norm The norm in C^*-algebras is useful for obtaining theoretical evaluations with milder dependencies on the data dimension than other norms. Indeed, for matrices, we can use the operator norm to alleviate the dependency of the generalization bound on the output data dimension, compared to the case where we regard a d by d matrix as a d^2-dimensional vector and use the vector norms such as the Euclidean norm (Hasimoto et al., 2023b).

Inner product We can naturally generalize the notion of inner product and Hilbert space by using C^*-algebra, which allows us to generalize RKHS to the space of C^*-algebra-valued functions. Learning C^*-algebra-valued maps is of great importance in many practical problems where the outputs to be predicted are not scalars but complex and structured data. The generalization of the notion of inner product and Hilbert space is by virtue of the properties of C^*-algebra, and this type of generalization is not easy for other notions than C^*-algebra. See Section 4 and Subsection 5.1 for more details.

4. C^*-algebra-valued Inner Product: The First Step in Constructing Algorithms

After representing data and models using C^*-algebras, we incorporate them into algorithms. In the algorithms, we generalize real- or complex-valued notions to C^*-algebra-valued ones. For methods implemented in Hilbert spaces, an important notion is the inner product. For example, a projection of a vector onto a low-dimensional subspace is obtained by computing inner products between the vector and vectors in an orthonormal basis of the subspace. In addition, for functions in RKHSs (Saitoh & Sawano, 2016; Hashimoto et al., 2021), the evaluation of a function at a point is obtained by computing the inner product of the function and the feature vector corresponding to the point (see Subsection 5.1 for details). To analyze structured data such as functional and graph data represented by a C^*-algebra, generalizing the inner product to the C^*-algebra enables us to extract more information than the standard complex-valued inner product.

The space that has the structure of the C^*-algebra-valued inner product is called Hilbert C^*-module (Lance, 1995), which is a generalization of Hilbert space. In the following, we review the definition of Hilbert C^*-module. We first introduce C^*-module over a C^*-algebra $\mathcal{A}$, which is a generalization of a vector space.

Definition 4.1 (C^*-module)**.** Let $\mathcal{M}$ be an abelian group with an operation $+$. If $\mathcal{M}$ is equipped with a (right) $\mathcal{A}$-multiplication, $\mathcal{M}$ is called a (right) C^*-*module* over $\mathcal{A}$.

We replace the vector space with a C^*-module to represent data. For example, we use $\mathbb{R}^d$ or $\mathbb{C}^d$ to represent d real- or complex-valued elements of a sample. If a sample is composed of d elements in $\mathcal{A}$, e.g., d functions, then we replace $\mathbb{R}^d$ or $\mathbb{C}^d$ with a C^*-module $\mathcal{A}^d$. Although we consider right multiplications in this paper, considering left multiplications instead of right multiplications is also possible.

In a C^*-module, we can consider an $\mathcal{A}$-valued inner product.

Definition 4.2 ($\mathcal{A}$-valued inner product)**.** A $\mathbb{C}$-linear map with respect to the second variable $\langle\cdot,\cdot\rangle_{\mathcal{M}} : \mathcal{M}\times\mathcal{M} \to \mathcal{A}$ is called an $\mathcal{A}$-valued *inner product* if it satisfies the following properties for $u, v, w \in \mathcal{M}$ and $c, d \in \mathcal{A}$:

1. $\langle u, vc + wd\rangle_{\mathcal{M}} = \langle u, v\rangle_{\mathcal{M}} c + \langle u, w\rangle_{\mathcal{M}} d$,
2. $\langle v, u\rangle_{\mathcal{M}} = \langle u, v\rangle^*_{\mathcal{M}}$,
3. $\langle u, u\rangle_{\mathcal{M}} \geq_{\mathcal{A}} 0$,
4. If $\langle u, u\rangle_{\mathcal{M}} = 0$ then $u = 0$.

Analogous to the case of C^*-algebra, we can define two notions to measure the magnitude of an element in a C^*-module $\mathcal{M}$ equipped with an $\mathcal{A}$-valued inner product.

Definition 4.3 ($\mathcal{A}$-valued absolute value and norm)**.** For $u \in \mathcal{M}$, the *$\mathcal{A}$-valued absolute value* $|u|_{\mathcal{M}}$ on $\mathcal{M}$ is defined by the positive element $|u|_{\mathcal{M}}$ of $\mathcal{A}$ such that $|u|^2_{\mathcal{M}} = \langle u, u\rangle_{\mathcal{M}}$. The (real-valued) norm $\|\cdot\|_{\mathcal{M}}$ on $\mathcal{M}$ is defined by $\|u\|_{\mathcal{M}} = \big\||u|_{\mathcal{M}}\big\|_{\mathcal{A}}$.

Definition 4.4 (Hilbert C^*-module)**.** Let $\mathcal{M}$ be a C^*-module over $\mathcal{A}$ equipped with an $\mathcal{A}$-valued inner product. If $\mathcal{M}$ is complete with respect to the norm $\|\cdot\|_{\mathcal{M}}$, it is called a *Hilbert C^*-module* over $\mathcal{A}$ or *Hilbert $\mathcal{A}$-module*.

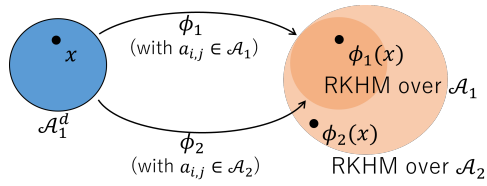

Figure 2. Overview of kernel methods with RKHMs by Hasimoto et al. (2023a). Here, $\mathcal{A}_1$ and $\mathcal{A}_2$ are C^*-algebras and $a_{i,j}$ is the parameter of the C^*-algebra-valued positive definite kernel associated with the feature maps ϕ_1 and ϕ_2. If $\mathcal{A}_1 \subseteq \mathcal{A}_2$, then the RKHM over $\mathcal{A}_1$ is contained in the RKHM over $\mathcal{A}_2$.

5. C^*-algebraic Kernel Methods and Neural Networks

We present two examples of machine learning methods, kernel methods and neural networks, to show how and why we apply C^*-algebra.

5.1. C^*-algebra and kernels: from Hilbert spaces to Hilbert modules

Reproducing kernel Hilbert spaces (RKHSs) enable us to extract nonlinear features of data (Schölkopf & Smola, 2001; Saitoh & Sawano, 2016). We first define a complex-valued function k, which is called positive definite kernel, and construct a Hilbert space called RKHS using k. Since RKHSs have high representation power and are theoretically solid, they have been applied to various machine learning methods, such as principal component analysis, support vector machine, and regression. However, RKHSs and its vector-valued generalization vvRKHSs are complex- and vector-valued function spaces; the output of the models are the usually complex- and vector-valued, respectively. Thus, we cannot represent functions whose outputs are in C^*-algebras. In addition, for structured data, complex-valued kernels degenerate the data to a complex value and extracting the information on the structure of data is difficult. Therefore, the construction of an appropriate positive definite kernel k is not easy. To resolve these issues, we generalize the positive definite kernel and RKHS by means of C^*-algebra (Heo, 2008; Hashimoto et al., 2022). Then, we can define reproducing kernel Hilbert C^*-module (RKHM), which is a generalization of RKHS. RKHMs are Hilbert C^*-modules. Thus, they have C^*-algebra-valued inner products. In addition, they are spaces of C^*-algebra-valued functions. By setting a suitable C^*-algebra, we can design a suitable RKHM for the given data. Figure 2 shows an overview of kernel methods with RKHMs. Applying RKHMs gives us a new twist on kernel methods.

We review the definition of RKHM below. First, we define an $\mathcal{A}$-valued positive definite kernel on a set $\mathcal{X}$ for data.

Definition 5.1 ($\mathcal{A}$-valued positive definite kernel)**.** An $\mathcal{A}$-valued map $k : \mathcal{X} \times \mathcal{X} \to \mathcal{A}$ is called a *positive definite kernel* if it satisfies the following conditions:

1. $k(x, y) = k(y, x)^*$ for $x, y \in \mathcal{X}$,
2. $\sum_{i,j=1}^n c_i^* k(x_i, x_j) c_j \geq_{\mathcal{A}} 0$ for $n \in \mathbb{N}$, $c_i \in \mathcal{A}$, $x_i \in \mathcal{X}$.

Hasimoto et al. (2023a) proposed to constructing C^*-algebra-valued kernels using the product structure of the C^*-algebra. They considered a kernel with circulant matrices and the product with matrix-valued parameters of the kernel, which enables us to use an operation that goes beyond the convolution.

Let $\phi : \mathcal{X} \to \mathcal{A}^{\mathcal{X}}$ be the *feature map* associated with k, which is defined as $\phi(x) = k(\cdot, x)$ for $x \in \mathcal{X}$. We construct the following C^*-module composed of $\mathcal{A}$-valued functions:

$$\mathcal{M}_{k,0} := \left\{ \sum_{i=1}^n \phi(x_i) c_i \,\middle|\, n \in \mathbb{N},\ c_i \in \mathcal{A},\ x_i \in \mathcal{X} \right\}.$$

Let $\langle \cdot, \cdot \rangle_{\mathcal{M}_k} : \mathcal{M}_{k,0} \times \mathcal{M}_{k,0} \to \mathcal{A}$ defined as

$$\left\langle \sum_{i=1}^n \phi(x_i) c_i, \sum_{j=1}^l \phi(y_j) d_j \right\rangle_{\mathcal{M}_k} := \sum_{i=1}^n \sum_{j=1}^l c_i^* k(x_i, y_j) d_j$$

for $c_i, d_i \in \mathcal{A}$ and $x_i, y_i \in \mathcal{X}$. By the properties in Definition 5.1 of k, $\langle \cdot, \cdot \rangle_{\mathcal{M}_k}$ is an $\mathcal{A}$-valued inner product and has the reproducing property

$$\langle \phi(x), v \rangle_{\mathcal{M}_k} = v(x)$$

for $v \in \mathcal{M}_{k,0}$ and $x \in \mathcal{X}$. Since v is an $\mathcal{A}$-valued function, this reproducing property enables us to deal with $\mathcal{A}$-valued functions, such as the regression of $\mathcal{A}$-valued functions (Hasimoto et al., 2023a).

The *reproducing kernel Hilbert $\mathcal{A}$-module (RKHM)* associated with k is defined as the completion of $\mathcal{M}_{k,0}$. We denote by $\mathcal{M}_k$ the RKHM associated with k.

An advantage of using C^*-algebras is that the operator norm is available. For the case of $\mathcal{A} = \mathbb{C}^{d \times d}$, we can also regard $\mathcal{A}$ as a Hilbert space equipped with the Hilbert–Schmidt inner product. However, in this case, the norm of a matrix $a \in \mathbb{C}^{d \times d}$ is calculated as $\sum_{i=1}^d \sum_{j=1}^d |a_{i,j}|^2$, where $a_{i,j}$ is the (i, j)-entry of a. On the other hand, the operator norm of a is calculated as $\max_{\|v\|=1} \sum_{i=1}^d |\sum_{j=1}^d a_{i,j} v_j|^2$. Since $|\sum_{j=1}^d a_{i,j} v_j| \leq \sum_{j=1}^d |a_{i,j}|$ and $\sum_{i=1}^d |v_j|^2 = 1$, the dependency of the operator norm on the dimension d is milder than that of the Hilbert–Schmidt norm. This fact is useful for deriving the generalization bound of the kernel ridge regression. By virtue of introducing C^*-algebra and using the operator norm, we can alleviate the dependency of the generalization bound on the output dimension (Hasimoto et al., 2023b).

5.1.1. Kernel mean embedding

Kernel mean embedding enables us to generalize kernel methods to analyze the distribution of data (Muandet et al.,

2017; Sriperumbudur et al., 2011). We define a map that maps a distribution to a vector in an RKHS by integrating the positive definite kernel with respect to the distribution. This map is called kernel mean embedding and enables us to analyze the distribution in the RKHS. We can generalize the kernel mean embedding using C^*-algebras (Hashimoto et al., 2022). Theoretically, to define the kernel mean embedding, we need the Riesz representation theorem. Although the Riesz representation theorem is always true for Hilbert spaces, we do not always have the corresponding theorem for Hilbert C^*-modules. According to Skeide (2000), we have the Riesz representation theorem if the Hilbert C^*-module is in a special class called von-Neumann module (see Definition 4.4 in Skeide 2000). In this case, instead of the $[0, 1]$-valued (more generally, finite signed or complex-valued) measure, we can map an $\mathcal{A}$-valued measure (Hashimoto et al., 2021) to a vector in an RKHM. $\mathcal{A}$-valued measures are defined as the special case of vector measures (Dinculeanu, 1967; 2000). Spectral measures and positive operator-valued measures are examples of $\mathcal{B}(\mathcal{W})$-valued measures for some Hilbert space $\mathcal{W}$. Positive operator-valued measures are introduced in quantum mechanics and are used in extracting information on the probabilities of outcomes from a state (Holevo, 2011). Using the kernel mean embedding for $\mathcal{A}$-valued measures, we can analyze positive operator-valued measures. Hashimoto et al. (2021) used the principal component analysis with kernel mean embedding to RKHM to analyze the strength of interaction effects for functional data. They also propose a MMD (maximal mean discrepancy) with kernel mean embedding to RKHM. Using the MMD with RKHM, we can define a distance between two positive operator-valued measures. We can also apply it to anomaly detection for quantum states (Hashimoto et al., 2020).

5.1.2. Deep learning with kernels

Combining kernel methods and deep learning to take advantage of the representation power and the theoretical solidness of kernel methods, and the flexibility of deep learning has been investigated (Cho & Saul, 2009; Gholami & Hajisami, 2016; Bohn et al., 2019; Laforgue et al., 2019). A generalization of these methods to RKHMs is also proposed (Hasimoto et al., 2023b). In this method, instead of considering the composition of functions in RKHSs, we consider the composition of functions in RKHMs. The high representation power of RKHMs and the product structure of C^*-algebras make the deep learning method with kernels more powerful.

5.2. Neural network parameters

In classical neural networks, the input and output are vectors whose elements are real or complex values. The learning parameters are also real or complex values. If the input and output are represented using C^*-algebras, then in some cases the corresponding parameters should also be C^*-algebra-valued. Hashimoto et al. (2022) proposed C^*-algebra net to combine multiple neural network models into a neural network with C^*-algebra-valued parameters. Figure 3 shows an overview of the C^*-algebra net. Using this new framework, we can combine multiple neural networks continuously, which is expected to be effective for ensemble, multitask, and meta-learning to fully extract features of data from multiple models or tasks. In addition, the experiment by Hashimoto et al. (2022) shows that this framework is useful for the case where the number of training samples is limited. As we stated in Section 3, we often come across these situations.

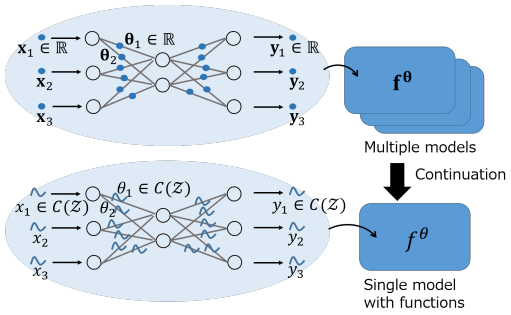

Figure 3. Overview of C^*-algebra net by Hashimoto et al. (2022). They focused on the C^*-algebra $C(\mathcal{Z})$ for a compact Hausdorff space $\mathcal{Z}$ and generalized neural network parameters to C^*-algebra-valued. We can continuously combine multiple (real-valued) neural networks using a single C^*-algebra net.

We review the technical details of C^*-algebra net. In this subsection, we focus on the case where $\mathcal{A}$ is the C^*-algebra $C(\mathcal{Z})$ for a compact Hausdorff space $\mathcal{Z}$. Let L be the number of layers, and for $j = 0, \dots, L$, let d_j be the width of the jth layer (d_0 is the input dimension). Let $W^j \in \mathcal{A}^{d_j \times d_{j-1}}$ and $b^j \in \mathcal{A}^{d_j}$ be the weight matrix and the bias of the jth layer, each of whose element is in $\mathcal{A}$. In addition, let $\sigma_j : \mathcal{A}^{d_j} \to \mathcal{A}^{d_j}$ be a nonlinear activation function. The C^*-algebra net f is defined as

$$f(x) = \sigma_L(W^L \sigma_{L-1}(\cdots \sigma_1(W^1 x + b^1) + \cdots) + b^L)$$

for $x \in \mathcal{A}^{d_0}$. If σ_j is pointwise, i.e., $\sigma_j(x)(z) = \tilde{\sigma}_j(x(z))$ for any $x \in \mathcal{A}^{d_j}$ and some $\tilde{\sigma}_j : \mathbb{C}^{d_j} \to \mathbb{C}^{d_j}$, then we have

$$f(x)(z) = \tilde{\sigma}_L(W^L(z)\tilde{\sigma}_{L-1}(\cdots \tilde{\sigma}_1(W^1(z)x(z) + b^1(z))\cdots) + b^L(z)). \quad (1)$$

Thus, we can represent infinitely many neural networks $\{f(x)(z)\}_{z \in \mathcal{Z}}$ by using a single C^*-algebra net. By learning the $\mathcal{A}$-valued parameters, we can learn the parameters of infinitely many neural networks simultaneously. In the following, we denote $f(x)(z)$ by $f_z(x)$.

A C^*-algebra net f provides infinitely many $\mathbb{C}^{d_L}$-valued networks f_z indexed by z. If we need a single $\mathbb{C}^{d_L}$-valued

network, we can integrate f over $\mathcal{Z}$. Assume $\mathcal{Z}$ is a measurable space and for any $x \in \mathcal{A}^{d_0}$, $z \mapsto f_z(x)$ is measurable. Let P be a probability measure on $\mathcal{Z}$. Consider a map $\tilde{f} : \mathcal{A}^{d_0} \to \mathbb{C}^{d_L}$ defined as

$$\tilde{f}(x) = \int_{\mathcal{Z}} f_z(x) \mathrm{d}P(z), \tag{2}$$

which is an ensemble of functions $\{f_z\}_{z \in \mathcal{Z}}$ with respect to the probability measure P.

In practical computations, we cannot deal with the infinite-dimensional space $\mathcal{A}$ itself. Thus, we restrict $\mathcal{A}$ to a finite-dimensional space. Let $\{v_1, \dots, v_m\}$ be a basis of the finite-dimensional space. We represent each element of the weights as $w^j_{i,k} = \sum_{l=1}^m c^j_{l,i,k} v_l$ with coefficients $c^j_{l,i,k} \in \mathbb{C}$. Here, $w^j_{i,k}$ is the (i,k)-element of the $\mathcal{A}^{d_j \times d_{j-1}}$-valued weight matrix W^j. Then, the jth layer f_j of the C^*-algebra net is represented as

$$(f_j(x))_i = \sigma_j \left(\sum_{k=1}^{d_{j-1}} \sum_{l=1}^{m} c^j_{l,i,k} v_l x_k^{j-1} + b^j_i \right) \in \mathcal{A}, \tag{3}$$

where $(f_j(x))_i$ is the ith element of the $\mathcal{A}^{d_j}$-valued vector $f_j(x) \in \mathcal{A}^{d_j}$. In addition, $x^0 = x$ is the input and $x^j = \sigma_j(W x^{j-1} + b^j)$ is the output of jth layer. In this case, the weight parameters of the jth layer of the C^*-algebra net are described by $m d_j d_{j-1}$ real- or complex-valued parameters.

In the following, we discuss advantages of C^*-algebra net. We first show new results about expressiveness and optimization. Then, we discuss existing investigations about interactions among models.

5.2.1. Expressiveness

An advantage of the C^*-algebra net is the expressiveness with respect to the variable z. We focus on a simple case for the discretized version of the C^*-algebra net (3) and show that the representation power of $f_z(x)$ grows as L grows even in this simple case. We will see that the C^*-algebra net is a polynomial of $v_1(z), \dots, v_l(z)$. The C^*-algebra net depends on z and x in different ways. It can be useful for the case where z and x have different attributions. For example, z is a time variable, and x is a space variable.

Assume the activation function σ_j is linear, that is, it satisfies $\sigma_j(\sum_{i=1}^m c_i u_i + b) = \sum_{i=1}^m \sigma_j(c_i) u_i + \sigma_j(b)$ for any $c_1, \dots, c_m, u_1, \dots, u_m, b \in \mathbb{C}$. In addition, for simplicity, we assume the input and the biases are constant functions, i.e., $x(z) = \hat{x}$ and $b^j(z) = \hat{b}^j$ for any $z \in \mathcal{Z}$, some $\hat{x} \in \mathbb{C}^{d_0}$, and some $\hat{b}^j \in \mathbb{C}^{d_j}$. We have the following proposition.

Proposition 5.2. *The C^*-algebra net $f_z(x)$ is a degree L polynomial with respect to $v_1(z), \dots, v_m(z)$.*

See Appendix B for the proof of Proposition 5.2. Proposition 5.2 shows that even if this simple case of the activation function σ_j is linear, $f_z(x)$ is nonlinear with respect to z. We can also construct a network whose input space is $\mathbb{C}^{d_0} \times \mathcal{Z}$. However, the situation of the C^*-algebra net with a finite-dimensional approximation is totally different from the case of the network on $\mathbb{C}^{d_0} \times \mathcal{Z}$. For the case of the network on $\mathbb{C}^{d_0} \times \mathcal{Z}$, if all the activation functions are the ReLU, defined as $\sigma(x) = \max\{0, x\}$ for $x \in \mathbb{R}$, then the approximation is obtained by the piecewise linear functions with respect to z (Hanin & Rolnick, 2019). On the other hand, in the case of the C^*-algebra net, if all the activation functions are linear, then the approximation is obtained by the polynomials with respect to $v_l(z)$. This fact means that even if the activation functions are linear, the representation power of the network with respect to z grows as L becomes large. In summary, the expressive power of C^*-algebra nets is high enough so that they can represent polynomials with respect to z even if the activation functions are linear. In addition, using C^*-algebra nets, we can induce a new type of nonlinearity that is different from the nonlinearity induced by the classical neural networks.

5.2.2. Optimization

Another advantage of the C^*-algebra net is that it fills the gap between convex and nonconvex optimization problems. Since the weight matrices of C^*-algebra nets are functions, they correspond to infinitely many weight matrices. Therefore, if we set a C^*-algebra appropriately, then we can represent an arbitrary scalar-valued network as a single C^*-algebra network. As a result, the optimization problem of the standard scalar-valued network is reduced to a convex optimization problem of a C^*-algebra network. Convex optimization of neural networks has been proposed and investigated (Bengio et al., 2005; Nitanda & Suzuki, 2017; Chizat & Bach, 2018; Daneshmand et al., 2023). In these studies, they consider learning the distribution of the weight parameters, which makes the optimization problem convex. On the other hand, the objective function becomes highly nonconvex if we consider optimizing weight parameters themselves, not the distribution of them. We show that the framework of C^*-algebra nets fills the gap between these convex and nonconvex optimizations.

For simplicity, we focus on neural networks with real-valued parameters and assume the input x is in the form $x(z) = \hat{x}$ for some $\hat{x} \in \mathbb{R}^{d_0}$. Let Ω be an interval in $\mathbb{R}$, and let $\mathcal{W}$ be a compact space. Let $\alpha^j_{i,k} : \mathcal{W} \to \Omega$ be a surjective function for $j = 1, \dots, L$, $i = 1, \dots, d_j$, and $k = 1, \dots, d_{j-1}$. The simplest example is setting $\mathcal{W} = \Omega$ and $\alpha^j_{i,k}$ as the identity. Let $N = \sum_{j=1}^L d_j(d_{j-1} + 1)$, the number of parameters (weight and bias parameters), $\mathcal{Z} = \mathcal{W}^N$, and $\mathcal{A} = C(\mathcal{Z})$. Define $W^j : \mathcal{Z} \to \mathbb{R}^{d_j \times d_{j-1}}$ and $b^j : \mathcal{Z}^N \to \mathbb{R}^{d_j}$ as

$$w^j_{i,k}(z^1_{1,1}, \dots, z^j_{i-1,k}, z, z^j_{i+1,k}, \dots, z^L_{d_L, d_{L-1}+1}) = \alpha^j_{i,k}(z)$$

$$b_i^j(z_{1,1}^1, \dots, z_{i-1,d_{j-1}+1}^j, z, z_{i+1,d_{j-1}+1}^j, \dots, z_{d_L,d_{L-1}+1}^L) = \alpha_{i,k}^j(z)$$

for any $z_{1,1}^1, \dots, z_{i,k-1}^j, z_{i,k+1}^j, \dots, z_{d_L,d_{L-1}+1}^L \in \mathcal{Z}$ and $z_{1,1}^1, \dots, z_{i-1,d_{j-1}+1}^j, z_{i+1,d_{j-1}+1}^j, \dots, z_{d_L,d_{L-1}+1}^L$, respectively. Let $\hat{\sigma} : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}$ be an activation function for standard scalar-valued networks. Set the activation function $\sigma_j : \mathcal{A}^{d_j} \to \mathcal{A}^{d_j}$ as $\sigma_j(x)(z) = \hat{\sigma}_j(x(z))$ for any $x \in \mathcal{A}^{d_j}$ and $z \in \mathcal{Z}$. With the above W^j, b^j, and σ_j, we construct the C^*-algebra net f. Then, we can show that we can represent any real-valued neural network by the form of f_z.

Let $\mathcal{F}$ be the class of the C^*-algebra nets defined above. Assume for any $\hat{x} \in \mathbb{R}^{d_0}$, $f(\hat{x})$ is bounded on $\mathcal{Z}$. Let $\mathcal{P}(\mathcal{Z})$ be the set of probability measures on $\mathcal{Z}$. As we mentioned as Eq. (2), we integrate $f \in \mathcal{F}$ over $\mathcal{Z}$ with respect to a probability measure $P \in \mathcal{P}(\mathcal{Z})$ to get a scalar-valued function. For $P \in \mathcal{P}(\mathcal{Z})$, let A_P be the integral operator on $\mathcal{F}$ defined as $A_P f(\hat{x}) = \int_{\mathcal{Z}} f_z(\hat{x}) \mathrm{d}P(z)$ for $\hat{x} \in \mathbb{R}^{d_0}$. The averaged C^*-algebra net $A_P f(\hat{x})$ is regarded as a continuation of the $(L+1)$-layer neural network $\sum_{i=1}^{d_{L+1}} p_i f_{z_i}(\hat{x})$, where $d_{L+1} \in \mathbb{N}$ and p_i is the weight parameter of the final layer. Figure 4 schematically shows the averaged C^*-algebra net $A_P f(\hat{x})$. We can see that the optimization problem of learning P is convex. See Appendix A for more details.

The cases 1) Fixing $f \in \mathcal{F}$ and optimizing P, and 2) optimizing a real-valued network f_z with respect to the weight parameters are two extremes. By virtue of the C^*-algebra net, we can define intermediate cases. Let $K \in \{0, \dots, N\}$ and let $M_1, \dots, M_K$ be disjoint subsets of $\mathcal{I} := \{(j,i,k) \mid j = 1, \dots, L,\ i = 1, \dots, d_j,\ k = 1, \dots, d_{j-1}\}$ and let $\mathcal{Z} = \mathcal{W}^K$. If $K = 0$, then we set $\mathcal{Z} = \mathcal{W}^0$ as a singleton, and we also set M_0 as an infinite set containing $\mathcal{I}$. Note that in the case where $\mathcal{Z}$ is a singleton, $C(\mathcal{Z})$ is isomorphic to $\mathbb{C}$. For $l = 1, \dots, K$, the variables $z_{i,k}^j$ for $(j,i,k) \in M_l$ are tied together and represented as a single variable. Let

$$w_{i,k}^j(z_1, \dots, z_{l-1}, z, z_{l+1}, \dots, z_K) = \alpha_{i,k}^j(z)$$

for $(j,i,k) \in M_l$. If $K = N$, then we reconstruct the case 1). If $K = 0$, then we have $w_{i,k}^j \in \mathbb{C}$, and $\alpha_{i,k}^j$ is necessarily a constant function. In addition, since $\mathcal{Z}$ is a singleton, $\mathcal{P}(\mathcal{Z}) = \{1\}$. As a result, we reconstruct the case 2). As for the learning, if $(j,k,i) \in M_l$ for some l with $|M_l| \geq 2$, then we learn $\alpha_{i,k}^j$. If $(j,i,k) \in M_l$ for some l with $|M_l| = 1$, then we fix $\alpha_{i,k}^j$. We also learn P.

5.2.3. Interactions among models

The product structure of the C^*-algebra gives the model additional structures. A generalization of the C^*-algebra net to noncommutative C^*-algebra is also proposed (Hataya & Hashimoto, 2023). In the above framework of the C^*-algebra network, we focus on the C^*-algebra of continuous functions, whose product structure is commutative. Therefore, if we consider the original C^*-algebra net (1), not the discretized one (3), then we have a separated neural network f_z for each z. The models do not interact without designing additional regularization terms to the loss function. By regarding $w \in C(\mathcal{Z})$ as the multiplication operator $M_w \in \mathcal{B}(L^2(\mathcal{Z}))$ defined as $M_w v = w \cdot v$, we can regard the C^*-algebra net over $C(\mathcal{Z})$ as that over $\mathcal{B}(L^2(\mathcal{Z}))$, where $L^2(\mathcal{Z})$ is the space of square-integrable functions on $\mathcal{Z}$. For the case where $\mathcal{Z}$ is a finite set, $C(\mathcal{Z})$ corresponds to the space of squared diagonal matrices, and $\mathcal{B}(L^2(\mathcal{Z}))$ corresponds to the space of general squared matrices. Replacing $C(\mathcal{Z})$ with $\mathcal{B}(L^2(\mathcal{Z}))$ and adding the nondiagonal part to the weight parameter $w_{i,k}^j$, we cannot separate each network f_z since the product structure becomes more complicated. Hataya & Hashimoto (2023) also proposed C^*-algebra nets over group C^*-algebras, which enable us to construct group equivariant neural networks by virtue of the noncommutative product structure in the group C^*-algebra.

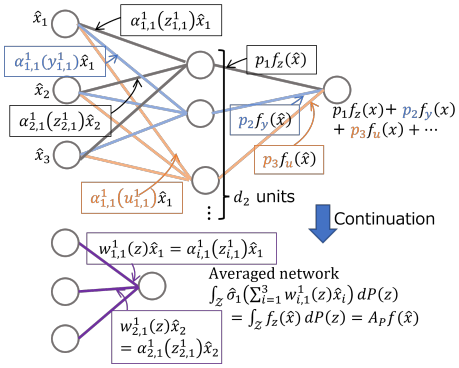

Figure 4. Overview of the averaged C^*-algebra net $A_P f(x)$. Here, $f_z(x) = \hat{\sigma}_1(\sum_{i=1}^3 \alpha_{i,1}^1(z) x_i)$. We can regard $A_P f(x)$ as a continuation of the 2-layer neural network $\sum_{i=1}^{d_2} p_i f_{z_i}(x)$.

6. Future Directions

As we discussed in the previous sections, generalizing machine learning methods by means of C^*-algebra enables us to go beyond the existing methods. However, there are many challenges involved. We discuss some them.

6.1. Challenges

Implementation and computational cost When we implement the methods with C^*-algebras, we have to represent elements in the C^*-algebras so that they are suitable for the computation. If the C^*-algebra is an infinite-dimensional space, we have to somehow discretize elements in the C^*-algebra. Hashimoto et al. (2021) use Fourier functions to discretize the functions in a C^*-algebra. Using kernel ridge regression to represent functions in a C^*-algebra is also proposed (Hashimoto et al., 2022). However, the effect of these methods on the entire algorithms has not been inves-

tigated, and representing elements in C^*-algebra properly is an important future direction of research. Even if we have an appropriate discretization method, the computational cost becomes expensive if the number of points for the discretization is large. For example, for kernel methods, if we represent elements in the C^*-algebra as d by d matrices, then the size of the Gram matrix is nd by nd, where n is the sample size. The cost for the computation involving the Gram matrix is expensive if n and d are large. In addition, for neural networks, if we represent weight parameters as d by d matrices, then the number of learnable parameters is d^2 times as large as that for the standard neural network with the same architecture. To alleviate the dependency on d, appropriate representations of elements of C^*-algebras to reduce computational costs should be investigated. In addition, although source codes are provided by the authors of the papers, as far as we know, no software for C^*-algebraic machine learning has been developed so far. The development of software is crucial to familiarize the machine learning community with the concept of C^*-algebra.

Lack of the inverse An element in a C^*-algebra does not always have its inverse. This situation is different from the standard complex- or real-valued case. This difference makes it difficult for us to generalize algorithms and theorems in Hilbert spaces straightforwardly. In fact, in Hilbert C^*-modules, once we normalize a vector, we cannot reconstruct the original vector exactly. However, we can obtain a normalized vector reconstructing a vector that is sufficiently close to the original vector (Hashimoto et al., 2023, Proposition 3.2). In addition, we only have an approximate version of the representer theorem for RKHMs (Hashimoto et al., 2023, Theorem 4.5). We sometimes have to give up constructing the exactly same algorithms and results as those in Hilbert spaces and devote ourselves to investigate how we can approximately obtain the algorithms and results.

Dealing with infinite-dimensional spaces Proving theoretical aspects of applying C^*-algebras to machine learning is not always straightforward. For example, as we mentioned in Subsection 5.1.1, Riesz representation theorem is not always true for Hilbert C^*-modules. In addition, for a submodule of a Hilbert C^*-module, its orthogonal complement does not always exist (Manuilov & Troitsky, 2000). Since these properties are fundamental for Hilbert spaces and used in proving and guaranteeing the theoretical aspects of machine learning methods, we have to be careful when we try to analyze methods with C^*-algebras theoretically.

Designing kernels Further investigation for designing positive definite kernels using C^*-algebras would be interesting. The kernel proposed by Hasimoto et al. (2023a), which we discussed in Subsection 5.1, is based on the convolution and is suitable for image data. Other kernels for other types of data, such as graphs and functions, should be investigated.

Theory of C^*-algebra nets Investigating generalization property and implicit regularization of the C^*-algebra net is an interesting future work. For example, it would be interesting to consider what types of C^*-algebra induce generalization or implicit regularization. In addition, understanding the standard real-valued neural networks and developing new methods regarding them through C^*-algebra net would also be interesting. For example, constructing an optimization method based on the observation in Subsection 5.2.2 to obtain a better solution is future work.

C^*-algebraic quantum machine learning We can represent various notions in quantum machine learning using C^*-algebra. For example, density matrices are represented by matrices. Analyzing quantum states using RKHMs or C^*-algebra nets is an interesting direction of research. In addition, quantum gates are represented by unitary matrices. Constructing or analyzing quantum circuits using C^*-algebra nets is also an interesting direction of research. C^*-algebras could also give rise to new machine learning methods which can be implemented more efficiently using a quantum computer.

6.2. Further examples of C^*-algebra for future work

Cuntz algebra Let $\mathcal{W}$ be a Hilbert space. Cuntz algebra $\mathcal{O}_n$ is a C^*-algebra generated by the isometries on $\mathcal{W}$, i.e., linear operators on $\mathcal{W}$ satisfying $S_i^* S_i = 1$ $(i = 1, \ldots, n)$ and $\sum_{i=1}^n S_i S_i^* = 1$, where 1 is the identity map on $\mathcal{W}$ (Cuntz, 1977). We can represent variable-length data, each of whose element is a discrete value. For example, we can set $S_1, \ldots, S_n$ as the dictionary of words and represent sentence as the product of the words from $S_1, \ldots, S_n$. We can also set $S_1, \ldots, S_n$ as nodes and represent paths using the product of the nodes from $S_1, \ldots, S_n$. Moreover, there are existing studies for applying Cuntz algebras to represent the filters in signal processing (Jorgensen, 2007).

Approximately finite dimensional C^*-algebra A C^*-algebra is referred to as approximately finite (AF) dimensional if it is the closure of an increasing union of finite dimensional subalgebras (Davidson, 1996). We can use AF C^*-algebras for representing data whose dimensions can vary, such as the adjacent matrices of social network graphs.

7. Conclusion

We proposed C^*-algebraic machine learning and discussed advantages and challenges of applying C^*-algebra to machine learning methods. We can represent structured data and multiple models using C^*-algebras, and by incorporating them into algorithms, we can fully extract features of data. C^*-algebra gives us a new twist on machine learning. We hope that our analysis will lead to greater attention to C^*-algebra in machine learning.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

Hachem Kadri is partially supported by grant ANR-19-CE23-0011 from the French National Research Agency. Masahiro Ikeda is partially supported by grant JPMJCR1913 from JST CREST.

References

Bengio, Y., Roux, N., Vincent, P., Delalleau, O., and Marcotte, P. Convex neural networks. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, 2005.

Bohn, B., Griebel, M., and Rieger, C. A representer theorem for deep kernel learning. *Journal of Machine Learning Research*, 20(64):1–32, 2019.

Bonawitz, K., Kairouz, P., McMahan, B., and Ramage, D. Federated learning and privacy: Building privacy-preserving systems for machine learning and data science on decentralized data. *ACM Queue*, 19(5):87–114, 2021.

Chizat, L. and Bach, F. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Proceedings of the 32nd Neural Information Processing Systems (NeurIPS)*, 2018.

Cho, Y. and Saul, L. Kernel methods for deep learning. In *Proceedings of the 23rd Conference on Neural Information Processing Systems (NIPS)*, 2009.

Cuntz, J. Simple C^*-algebras generated by isometrie. *Communications in Mathematical Physics*, 57:173–185, 1977.

Daneshmand, H., Lee, J. D., and Jin, C. Efficient displacement convex optimization with particle gradient descent. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Davidson, K. R. *C^*-Algebras by Example*. American Mathematical Society, 1996.

Dinculeanu, N. *Vector Measures*. International Series of Monographs in Pure and Applied Mathematics ; Volume 95. Pergamon Press, Oxford, England, 1967.

Dinculeanu, N. *Vector Integration and Stochastic Integration in Banach Spaces*. John Wiley & Sons, New York, 2000.

Dong, X., Yu, Z., Cao, W., Shi, Y., and Ma, Q. A survey on ensemble learning. *Frontiers of Computer Science*, 14: 241–258, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., and Suganthan, P. N. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115: 105151, 2022.

Gholami, B. and Hajisami, A. Kernel auto-encoder for semi-supervised hashing. In *Proceedings of 2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

Hanin, B. and Rolnick, D. Complexity of linear regions in deep networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Hashimoto, Y., Ishikawa, I., Ikeda, M., Komura, F., and Kawahara, Y. Kernel mean embeddings of von Neumann-algebra-valued measures. arXiv:2007.14698, 2020.

Hashimoto, Y., Ishikawa, I., Ikeda, M., Komura, F., Katsura, T., and Kawahara, Y. Reproducing kernel Hilbert C^*-module and kernel mean embeddings. *Journal of Machine Learning Research*, 22(267):1–56, 2021.

Hashimoto, Y., Wang, Z., and Matsui, T. C^*-algebra net: A new approach generalizing neural network parameters to C^*-algebra. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Hashimoto, Y., Komura, F., and Ikeda, M. Hilbert C^*-module for analyzing structured data. In *Matrix and Operator Equations and Applications*, pp. 633–659. Springer Nature, 2023.

Hasimoto, Y., Ikeda, M., and Kadri, H. Learning in RKHM: a C^*-algebraic twist for kernel machines. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023a.

Hasimoto, Y., Ikeda, M., and Kadri, H. Deep learning with kernels through RKHM and the Perron-Frobenius operator. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023b.

Hataya, R. and Hashimoto, Y. Noncommutative C^*-algebra net: Learning neural networks with powerful product structure in C^*-algebra. arXiv: 2302.01191, 2023.

Heo, J. Reproducing kernel Hilbert C^*-modules and kernels associated with cocycles. *Journal of Mathematical Physics*, 49(10):103507, 2008.

Holevo, A. S. *Probabilistic and Statistical Aspects of Quantum Theory*. Monographs (Scuola Normale Superiore) ; 1. Scuola Normale Superiore, Pisa, 2011.

Jorgensen, P. E. *Analysis and Probability: Wavelets, Signals, Fractals*. Springer New York, 2007.

Kadri, H., Duflos, E., Preux, P., Canu, S., Rakotomamonjy, A., and Audiffren, J. Operator-valued kernels for learning from functional response data. *Journal of Machine Learning Research*, 17(20):1–54, 2016.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89), 2023.

Laforgue, P., Clémençon, S., and d'Alche Buc, F. Autoencoding any data through kernel autoencoders. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Lance, E. C. *Hilbert C^*-modules – a Toolkit for Operator Algebraists*. London Mathematical Society Lecture Note Series, vol. 210. Cambridge University Press, 1995.

Manuilov, V. and Troitsky, E. Hilbert C^*- and W^*-modules and their morphisms. *Journal of Mathematical Sciences*, 98:137–201, 2000.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10(1–2), 2017.

Murphy, G. J. *C*-Algebras and Hilbert Space Operators*. Academic Press, 1990.

Nitanda, A. and Suzuki, T. Stochastic particle gradient descent for infinite ensembles. arXiv:1712.05438, 2017.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. Latent multi-task architecture learning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Saitoh, S. and Sawano, Y. *Theory of reproducing kernels and applications*. Springer Singapore, 2016.

Schölkopf, B. and Smola, A. J. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press, Cambridge, MA, USA, 2001.

Skeide, M. Generalised matrix C^*-algebras and representations of Hilbert modules. *Mathematical Proceedings of the Royal Irish Academy*, 100A(1):11–38, 2000.

Sriperumbudur, B. K., Fukumizu, K., and Lanckriet, G. R. G. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12 (70):2389–2410, 2011.

Wang, J.-L., Chiou, J.-M., and Müller, H.-G. Functional data analysis. *Annual Review of Statistics and Its Application*, 3:257–295, 2016.

Zhang, Z., Luo, P., Loy, C. C., and Tang, X. Facial landmark detection by deep multi-task learning. In *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, 2014.

Zhu, B., Song, P., and Taylor, J. Stochastic functional data analysis: a diffusion model-based approach. *Biometrics*, 67(4):1295–1304., 2011.

Appendix

A. Details of Subsection 5.2.2

We provide the details of Subsection 5.2.2. With W^j, b^j, and σ_j defined in Subsection 5.2.2, let

$$f_j(x) = \sigma_j(W^j x + b^j)\ (j = 1, \ldots, L). \tag{4}$$

We consider the set of C^*-algebra nets defined as $\mathcal{F}_{\mathcal{Z}} = \{f_z \mid f = f_L \circ \cdots \circ f_1 \text{ with } f_j \text{in Eq. (4)}, z \in \mathcal{Z}\}$. In addition, we set the following function class of the standard real-valued networks:

$$\hat{\mathcal{F}} = \{\hat{f}_L \circ \cdots \circ \hat{f}_1 \mid \hat{f}_j(x) = \hat{\sigma}_j(\hat{W}^j x + \hat{b}^j),\ \hat{W}^j \in \Omega^{d_j \times d_{j-1}},\ \hat{b}^j \in \Omega^{d_j}\}.$$

Proposition A.1. *We have $\mathcal{F}_{\mathcal{Z}} = \hat{\mathcal{F}}$ as sets.*

Proof. Let $f_z \in \mathcal{F}_{\mathcal{Z}}$ for some $z \in \mathcal{Z}$. Let $\hat{w}^j_{i,k} = w^j_{i,k}(z)$ and $\hat{b}^j_i = b^j_i(z)$ for any $j = 1, \ldots, L$, $i = 1, \ldots, d_j$, and $k = 1, \ldots, d_{j-1}$. In addition, let $\hat{\sigma}_j(x) = \sigma_j(x 1_{\mathcal{A}})(z)$ for $x \in \mathbb{R}^{d_j}$, where $1_{\mathcal{A}}$ is the constant map defined as $1_{\mathcal{A}}(z) = 1$ for any $z \in \mathcal{Z}$. We construct $\hat{f} = \hat{f}_L \circ \cdots \circ \hat{f}_1$ as $\hat{f}_j(x) = \hat{\sigma}_j(\hat{W}^j x + \hat{b}^j)$ with $\hat{W}^j$, $\hat{b}^j$, and $\hat{\sigma}_j$ defined above. Then, we have $f_z = \hat{f}$ and $f_z \in \hat{\mathcal{F}}$. The converse is trivial by the definition of $\mathcal{F}_{\mathcal{Z}}$. □

Let $\mathcal{F}$ be the class of the functions $f = f_L \circ \cdots \circ f_1$ defined as Eq. (4). Assume for any $\hat{x} \in \mathbb{R}^{d_0}$, $f(\hat{x})$ is bounded on $\mathcal{Z}$. Let $\mathcal{P}(\mathcal{Z})$ be the set of probability measures on $\mathcal{Z}$. As we mentioned as Eq. (2), we integrate $f \in \mathcal{F}$ over $\mathcal{Z}$ with respect to a probability measure $P \in \mathcal{P}(\mathcal{Z})$ to get a scalar-valued function. For $P \in \mathcal{P}(\mathcal{Z})$, let A_P be the integral operator on $\mathcal{F}$ defined as $A_P f(\hat{x}) = \int_{\mathcal{Z}} f_z(\hat{x}) \mathrm{d}P(z)$ for $\hat{x} \in \mathbb{R}^{d_0}$. The following proposition shows the optimization problem of learning P is convex.

Proposition A.2. *Let $\mathcal{L} : \mathbb{R}^{d_L} \times \mathbb{R}^{d_L} \to \mathbb{R}_+$ be a loss function that is continuous and where for any $y \in \mathbb{R}^{d_L}$, the function $\mathcal{L}(\cdot, y)$ on $\mathbb{R}^{d_L}$ is convex. Fix f as a network defined as (4). Then, for any $\hat{x} \in \mathbb{R}^{d_0}$ and any $y \in \mathbb{R}^{d_L}$, the map $\mathcal{P}(\mathcal{Z}) \to \mathbb{R}_+$ defined as $P \mapsto \mathcal{L}(A_P f(\hat{x}), y)$ is convex.*

Proof. Since $\mathcal{L}(\cdot, y)$ is convex, for $P, Q \in \mathcal{P}(\mathcal{Z})$ and $t \in [0, 1]$, we have

$$\begin{aligned}\mathcal{L}(A_{tP+(1-t)Q} f(x), y) &= \mathcal{L}(tA_P f(x) + (1-t)A_Q f(x), y) \\ &\leq t\mathcal{L}(A_P f(x), y) + (1-t)\mathcal{L}(A_Q f(x), y),\end{aligned}$$

which completes the proof of the proposition. □

Remark A.3. Let $P = \delta_z$, where δ_z is the Dirac measure centered at $z \in \mathcal{Z}$. Then, $A_{\delta_z} f = f_z$. Proposition A.1 implies that learning z of f_z for a C^*-algebra network f corresponds to learning the weights $\hat{W}^j$ and biases $\hat{b}^j$ of the scalar-valued network $\hat{f}$. Note that the set $\{A_{\delta_z} \mid z \in \mathcal{Z}\}$ is not convex. By expanding the search space to the set of probability measures, the optimization problem becomes convex.

B. Proof of Proposition 5.2

Proposition 5.2 *The C^*-algebra net $f_z(x)$ is a degree L polynomial with respect to $v_1(z), \ldots, v_m(z)$.*

Proof. The k_Lth element of the output of $f_z(x)$ is written as

$$\begin{aligned}&(f_z(x))_{k_L} \\ &= \sigma_L\Bigg(\sum_{k_{L-1}=1}^{d_{L-1}} \sum_{l_L=1}^{m} c^L_{l_L,k_L,k_{L-1}} v_{l_L}(z) \sigma_{L-1}\Bigg(\cdots \sigma_2\Bigg(\sum_{k_1=1}^{d_1} \sum_{l_2=1}^{m} c^2_{l_2,k_2,k_1} v_{l_2}(z) \sigma_1\Bigg(\sum_{k_0=1}^{d_0} c^1_{l_1,k_1,k_0} v_{l_1}(z) \hat{x}_{k_0} \\ &\quad + \hat{b}^1_{k_1}\Bigg) + \hat{b}^2_{k_2}\Bigg)\cdots\Bigg) + \hat{b}^L_{k_L}\Bigg)\end{aligned}$$

$$
\begin{aligned}
&= \sum_{l_1,\ldots,l_L=1}^{m} v_{l_L}(z)\cdots v_{l_1}(z)\sigma_L\Bigg(\sum_{k_{L-1}=1}^{d_{L-1}} c^L_{l_L,k_L,k_{L-1}}\sigma_{L-1}\Bigg(\cdots\sigma_1\Bigg(\sum_{k_0=1}^{d_0} c^1_{l_1,k_1,k_0}\hat{x}_{k_0}\Bigg)\cdots\Bigg)\\
&\quad+\sum_{l_2,\ldots,l_L=1}^{m} v_{l_L}(z)\cdots v_{l_2}(z)\sigma_L\Bigg(\sum_{k_{L-1}=1}^{d_{L-1}} c^L_{l_L,k_L,k_{L-1}}\sigma_{L-1}\Bigg(\cdots\sigma_2\Bigg(\sum_{k_1=1}^{d_1} c^2_{l_2,k_2,k_1}\sigma_1(\hat{b}^1_{k_1})\Bigg)\cdots\Bigg)\\
&\quad+\sum_{l_L=1}^{m} v_{l_L}(z)\sigma_L\Bigg(\sum_{k_{L-1}=1}^{d_{L-1}} c^L_{l_L,k_L,k_{L-1}}\sigma_{L-1}(\hat{b}^{L-1}_{k_{L-1}})\Bigg)+\sigma_L(\hat{b}^L_{k_L}),
\end{aligned}
$$

which completes the proof of the proposition. □