Vol.2017-MPS-113 No.11
Vol.2017-BIO-50 No.11
2017/6/23

Randomized Kernel Mean Networks for Bag-of-Words Data

Yuya Yoshikawa[1,a)] Tomoharu Iwata[2,b)]

Abstract: In various machine learning problems, bag-of-words (BoW) representation, i.e., a multiset of features, is widely used as a simple and general data representation. Deep learning is successfully used in many areas. However, with BoW data, deep learning models are often outperformed by kernel methods such as support vector machines (SVMs), where each sample is simply transformed into a fixed-length count vector for the input. In this paper, we propose a deep learning model for BoW data. Based on the idea introduced in the framework of SVMs that has achieved a better performance in BoW count vector inputs, the proposed model assigns each feature to a latent vector, and each sample is represented by a distribution of the latent vectors of features contained in the sample. To transform the distribution efficiently and nonparametrically to the inputs of deep learning, we integrate kernel mean embeddings and a random Fourier feature algorithm. Our experiments verify the effectiveness of the proposed model on BoW document datasets. Because the proposed model is a general framework for BoW data, it can be applied directly to various supervised and unsupervised learning tasks. Moreover, because the proposed model can be combined with existing deep learning models, it further extends the potential applications of deep learning.

1. Introduction

In various machine learning problems, bag-of-words (BoW) representation, i.e., a multiset of features, is widely used as a simple and general data representation[*1]. In natural language processing, BoW is used to represent a document in where each feature corresponds to a word appearing in the document. Although a document can be treated as a word sequence, there are many cases where the original word sequences are unavailable and only BoW data are available because the original data contain privacy information, public datasets have been already preprosessed, and some words with unnecessary part-of-speech tags are removed. In bioinformatics, BoW is used to represent a protein structure, where each feature is a substructure of the protein structure [16]. Other examples include BoW of tags and key-phrases to search objects [20] and BoW of individuals to represent groups in group recommendation [1].

In this study, we propose a deep learning model for BoW data. Deep learning is successfully used in many areas, such as image recognition, machine translation and automatic game playing [6, 21, 5]. Deep learning allows extracting effective representations from complicated data, such as images and game of Go [17], because of its high flexibility. However, with BoW data, deep learning models are often outperformed by kernel methods such as support vector machines (SVMs) [10], where each sample is simply transformed into a fixed-length count vector for the input. Therefore, an approach to handle the BoW data effectively is needed to improve the deep learning performance.

The proposed method assumes that each feature is associated with a latent vector, and each sample is represented by the distribution of the latent vectors of features contained in the sample. The concept underlying this method was first introduced in the framework of SVMs [25], and it was shown that SVMs using this concept achieve better performance than those with BoW count vector inputs. Input uncertainty is incorporated by representing a sample as a distribution, and the relationship between features is captured by the latent vectors. We employ kernel mean embeddings [18] to represent distributions nonparametrically, where a distribution is represented as a point in a reproducing kernel Hilbert space (RKHS), (which is known as the kernel mean) without the need to define parametric distributions. A kernel mean embedding can be directly incorporated into kernel methods, such as SVMs [25] and Gaussian processes [26]. However, it cannot be directly incorporated into deep learning because the explicit value of the kernel mean is unavailable; it is indirectly accessible via inner product. With the proposed method, a random Fourier feature (RFF) algorithm [14] is used to obtain explicit values of the kernel mean efficiently. In particular, first, we obtain randomized feature vectors, which are approximations of the latent vectors in the RKHS, by applying the RFF algorithm to each of the latent vectors. Then, we compute the mean of the randomized feature vectors associated with each sample. The mean vector is an approximation of the kernel mean of the sample, and it preserves all

[1] Software Technology and Artificial Intelligence Research Laboratory, Chiba Institute of Technology, 2-17-1, Tsudanuma, Narashino, Chiba 275–0016, Japan
[2] NTT Communication Science Laboratories, 2-4, Hikaridai, Seika-cho, Kyoto 619–0237, Japan
[a)] yoshikawa@stair.center
[b)] iwata.tomoharu@lab.ntt.co.jp

[*1] Note that BoW data are also referred to as histogram over features and count data.

the moment information such as the mean, covariance, and higher-order moments, when characteristic kernels and enough length of randomized vectors are used. The mean vector is the input for deep learning such as multilayer perceptron (MLP), and predict the label as the output of the deep learning. We call the proposed method Randomized Kernel Mean Networks (RKM-Nets). In RKM-Nets, we estimate the latent vectors associated with the features and weight parameters for deep learning. The learning of these parameters can be performed efficiently through a standard backpropagation with stochastic gradient descent.

The method obtaining the randomized kernel mean can be regarded as a type of pooling methods. Instead of the randomized kernel mean, average pooling, i.e., calculating the mean of the latent vectors, can be used. However, average pooling loses covariance and high-order moments of the distribution for the latent vectors. Thus, by using the randomized kernel mean, RKM-Nets sufficiently capture the characteristics of the given BoW data and propagate them to next layers.

Because RKM-Nets are a general framework for BoW data, they can be applied directly to various supervised and unsupervised learning tasks. For example, by changing the MLP in RKM-Nets so as to output BoW count vectors, an auto-encoder model, an unsupervised representation learning method, can be produced. Moreover, because RKM-Nets are formulated as neural networks and implemented on deep learning frameworks, their extensions can be easily developed[*2]. For example, new models can be developed by concatenating RKM-Nets with other deep learning models such as convolutional neural networks. Such models can then be used, for example, in multimodal learning with BoW documents and images, and with BoW items purchased by users and their life logs. We believe that this study extends the potential applications of deep learning that contain BoW data.

2. Related Work

The present study is inspired by the work of Yoshikawa et al. which proposed a SVM-based discriminative model for BoW data based on kernel mean embeddings [25]. Although their method achieved state-of-the-art classification accuracy for BoW data, it had two drawbacks. First, the method is computationally quite expensive because it requires to calculate a Gram matrix of $O(n^2w^2)$ time complexity for each iteration where n is the number of samples and w is the average number of features contained in the samples. Thus, it cannot be applied to practical large-scale datasets like those used in our experiments. Second, although their idea can be applied to various machine learning tasks, e.g., [27], in order to develop a new model based on that idea, one needs to define a new formulation and derive a new parameter estimation method. This is a quite complicated and time-consuming process. Our work overcomes these drawbacks using an RFF algorithm and formulating it as a deep learning model, while inheriting the advantages of their aforementioned previous method.

For document and sentence data that retain the word order, recurrent neural networks and convolutional neural networks (CNN) that can model the word order were proposed by many studies [10, 8, 22, 7]. However, as we stated in Section 1, in many cases, only BoW data are provided in practice, thus, these models cannot be applied for BoW data.

Moreover, there are no studies on deep learning models specific for BoW data. In previous studies using BoW data in deep learning models, the data is converted to BoW count vectors and associated with fully-connected networks. In such cases, it was experimentally shown that the performance of the deep learning models is as good as or worth than that of kernel methods such as SVM [10]. RKM-Nets overcome the weakness of the previous deep learning models for BoW by incorporating techniques successful in kernel methods into deep learning.

In a few previous studies, kernel mean embeddings and random Fourier feature algorithms were applied to deep learning models. Dziugaite et al. and Li et al. used maximum mean discrepancy, an application of kernel mean embeddings, to train deep generative networks that generate true samples from noisy samples [3, 11]. Oliva et al. used both kernel mean embeddings and random Fourier features to capture the high-level features of images in CNN efficiently [13]. Our work is different from the existing studies since we employs these techniques for improving performance of deep learning with BoW data.

3. Preliminaries

In this section, we introduce two key techniques in RKM-Nets: the RFF algorithm and kernel mean embeddings.

3.1 Random Fourier Features

We introduce a RFF algorithm proposed in [14], which is used for approximating kernel values. The RFF algorithm is used for approximating shift-invariant kernels such as Gaussian RBF, Laplacian and Cauchy kernels.

Let $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$ be data points, and $k(\mathbf{x}, \mathbf{x}')$ be a shift-invariant kernel between $\mathbf{x}$ and $\mathbf{x}'$. The RFF algorithm allows us to obtain a random feature map $\mathbf{z} : \mathbb{R}^D \to \mathbb{R}^L$ such that $k(\mathbf{x}, \mathbf{x}') \approx \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{x}')$.

According to Bochner's theorem [15], the kernel $k(\mathbf{x}, \mathbf{x}')$ can be written as a Fourier transform as follows:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \int_{\mathbb{R}^D} p(\mathbf{r}) \exp\left(j\mathbf{r}^\top(\mathbf{x} - \mathbf{x}')\right) d\mathbf{r} \\ &= \mathbb{E}_{\mathbf{r}}\left[\zeta_{\mathbf{r}}(\mathbf{x})\zeta_{\mathbf{r}}(\mathbf{x}')^*\right], \end{aligned} \tag{1}$$

where j indicates a complex number, and we placed $\zeta_{\mathbf{r}}(\mathbf{x}) = \exp(j\mathbf{r}^\top\mathbf{x})$. Because both the probability distribution $p(\mathbf{r})$ and the kernel $k(\mathbf{x}, \mathbf{x}')$ are real, the complex exponentials in the integral (1) can be replaced with cosines. Therefore, instead of $\zeta_{\mathbf{r}}(\mathbf{x})$, we consider using $z(\mathbf{x}; \mathbf{r}, s) = \cos(\mathbf{r}^\top\mathbf{x} + s)$, where $\mathbf{r} \in \mathbb{R}^D$ and $s \in [0, 2\pi]$ are random variables generated from the distribution p and an uniform distribution. Therefore, (1) can be rewritten as $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{r},s}[z(\mathbf{x}; \mathbf{r}, s)z(\mathbf{x}'; \mathbf{r}, s)]$. The expectation is calculated by Monte Carlo approximation with L samples as follows:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{\mathbf{r},s}[z(\mathbf{x}; \mathbf{r}, s)z(\mathbf{x}'; \mathbf{r}, s)] \\ &\approx \frac{1}{L}\sum_{l=1}^{L} z(\mathbf{x}; \mathbf{r}_l, s_l)z(\mathbf{x}'; \mathbf{r}_l, s_l) \\ &= \mathbf{z}(\mathbf{x}; \mathbf{R}, \mathbf{s})^\top \mathbf{z}(\mathbf{x}'; \mathbf{R}, \mathbf{s}), \end{aligned} \tag{2}$$

[*2] The implementation will be open when this paper is published.

Algorithm 1 Constructing random features for Gaussian RBF kernel

Require: Input vector $\mathbf{x} \in \mathbb{R}^D$, bandwidth parameter for Gaussian RBF kernel $\gamma > 0$, random feature length L.

Ensure: random features $\mathbf{z}$ such that $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2^2/2\right) \approx \mathbf{z}(\mathbf{x}; \mathbf{R}, \mathbf{s})^\top \mathbf{z}(\mathbf{x}'; \mathbf{R}, \mathbf{s})$

1: **for** l in L **do**
2: Generate random variable $\mathbf{r}_l = \sqrt{2\gamma}\mathbf{a}_l$ where $\mathbf{a}_l \sim \mathcal{N}(\mathbf{0}_D, \mathbf{I}_D)$.
3: Generate random variable $s_l \sim \text{Uniform}(0, 2\pi)$
4: Construct random feature $f_l = z(\mathbf{x}; \mathbf{r}_l, s_l) = \sqrt{2}\cos(\mathbf{r}_l^\top \mathbf{x} + s_l)$
5: **end for**
6: **return** $\mathbf{z}(\mathbf{x}; \mathbf{R}, \mathbf{s}) = \sqrt{\frac{1}{L}}[f_1, f_2, \cdots, f_L]^\top$ where $\mathbf{R} = \{\mathbf{r}_l\}_{l=1}^L$ and $\mathbf{s} = \{s_l\}_{l=1}^L$

where $\mathbf{z}(\mathbf{x}; \mathbf{R}, \mathbf{s})$ are known as the *random features* of $\mathbf{x}$ and is defined as

$$\mathbf{z}(\mathbf{x}; \mathbf{R}, \mathbf{s}) = \frac{1}{\sqrt{L}}[z(\mathbf{x}; \mathbf{r}_1, s_1), z(\mathbf{x}; \mathbf{r}_2, s_2), \cdots, z(\mathbf{x}; \mathbf{r}_L, s_L)]^\top, \quad (3)$$

where $\mathbf{R} = \{\mathbf{r}_l\}_{l=1}^L$ and $\mathbf{s} = \{s_l\}_{l=1}^L$. In particular, the RFF algorithm for the Gaussian RBF kernel is shown in Algorithm 1. By using the random features $\mathbf{z}(\mathbf{x}; \mathbf{R}, \mathbf{s})$ in various machine learning problems instead of original data point $\mathbf{x}$, it is possible to produce nonlinear models while maintaining the formulation of linear ones, which incur much smaller computational costs than directly learning their corresponding nonlinear extended models.

3.2 Kernel Mean Embeddings

Kernel mean embeddings are used to embed any probability distribution P on a space $\mathcal{X}$ into a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ specified by kernel k, and the distribution is represented as an element $\mu^*(P)$ in the RKHS. $\mu^*(P)$ is also known as a *kernel mean*. More precisely, given a distribution P, the kernel mean $\mu^*(P)$ is defined as follows:

$$\mu^*(P) := \mathbb{E}_{\mathbf{x}\sim P}[k(\cdot, \mathbf{x})] = \int_{\mathcal{X}} k(\cdot, \mathbf{x}) dP \in \mathcal{H}_k, \quad (4)$$

where kernel k is referred to as the *embedding kernel*. Indeed, kernel mean $\mu^*(P)$ preserves the properties of the probability distribution P including the mean, covariance and higher-order moments by using *characteristic kernels* (e.g., Gaussian RBF kernel) [19].

When a set of samples $\mathbf{X} = \{\mathbf{x}_s\}_{s=1}^n$ is drawn from the distribution and by interpreting sample set $\mathbf{X}$ as the empirical distribution $\hat{P} = \frac{1}{n}\sum_{s=1}^n \delta_{\mathbf{x}_s}(\cdot)$, where $\delta_{\mathbf{x}}(\cdot)$ is the Dirac delta function at point $\mathbf{x} \in \mathcal{X}$, the empirical kernel mean $\mu(\mathbf{X})$ is given by

$$\mu(\mathbf{X}) = \frac{1}{n}\sum_{s=1}^n k(\cdot, \mathbf{x}_s), \quad (5)$$

which can be approximated with an error rate of $\|\mu(\mathbf{X}) - \mu^*(P)\|_{\mathcal{H}_k} = O_p(n^{-\frac{1}{2}})$ [18]. Unlike kernel density estimation, the error rate of the kernel embeddings is independent of the dimensionality of the given distribution.

The kernel mean embeddings technique was previously used for distribution data. The examples include statistical test of independence for two distributions [4] and discriminative learning for distributions [12].

4. Randomized Kernel Mean Networks

We propose that randomized kernel mean networks (RKM-Nets) are effective for modeling BoW data. RKM-Nets are formulated as a deep neural network incorporating the techniques of kernel methods described in Section 3 for representing BoW data. In order to demonstrate the effectiveness of the proposed method, we only consider classification problems, although it can be directly applied to regression problems as well.

Suppose that we are given training data $\mathcal{D}_{\text{tr}} = \{x_i, t_i\}_{i=1}^{N_{\text{tr}}}$, where $x_i = (\mathcal{A}_i, m_i)$ is a multi-set of the features associated with the ith training sample and $t_i \in \{1, 2, \cdots, C\}$ denotes its corresponding label indicator. Here, $\mathcal{A}_i \subseteq \mathcal{V}$ is a set of features included in a unique feature set (or vocabulary set) $\mathcal{V}$, and $m_i : \mathcal{A}_i \to \mathbb{R}_+$ denotes the multiplicity (or frequency) function[*3]. For example, $m_i(s)$ denotes the multiplicity of feature s in the ith sample.

4.1 The Model Architecture

Figure 1 illustrates the overall architecture of an RKM-Net for classification. The input to be RKM-Net is a multiset of features associated with the ith sample $x_i = (\mathcal{A}_i, m_i)$.

In an RKM-Net, each feature $s \in \mathcal{V}$ is represented as a latent vector $\mathbf{v}_s \in \mathbb{R}^{D_{\text{emb}}}$. Then, each sample is represented as a multiset of the latent vectors of the features associated with the sample. We consider this multiset of the latent vectors as samples from an unknown distribution. To capture the characteristics of the distribution from the multiset effectively and nonparametrically, we employ kernel mean embeddings technique, as described in Section 3.2. However, we do not use the kernel mean (5) directly, because the kernel mean cannot be obtained explicitly. Instead, we calculate an approximated kernel mean by using the RFF algorithm described in Section 3.1. In particular, for the latent vector $\mathbf{v}_s$, we construct the random features $\mathbf{z}_s \in \mathbb{R}^{D_{\text{rff}}}$ by applying Algorithm 1. Then, we calculate the weighted mean of the random features associated with the ith sample as follows:

$$\tilde{\mu}_i = \tilde{\mu}_{\mathbf{R},\mathbf{s}}(x_i; \mathbf{V}) = \frac{1}{\#x_i}\sum_{s\in\mathcal{A}_i} m_i(s) \cdot \mathbf{z}(\mathbf{v}_s; \mathbf{R}, \mathbf{s}), \quad (6)$$

where, $\mathbf{V} = \{\mathbf{v}_s\}_{s\in\mathcal{V}}$ and $\#x_i$ denotes the cardinality of multi-set x_i, that is, $\#x_i = \sum_{s\in\mathcal{A}_i} m_i(s)$. We refer to $\tilde{\mu}_i$ as a *randomized kernel mean* for the ith sample. The randomized kernel mean approximately holds the moment information about the distribution of the latent vectors. Because the randomized kernel mean is represented as a vector explicitly, it can be treated as input in existing deep neural networks. In this study, we employ MLP, which is one of the simplest deep neural networks. Here the MLP has a hidden layer with D_{rff} units. Finally, the MLP outputs the label confidence $\mathbf{y}_i = [y_{i1}, y_{i2}, \cdots, y_{iC}]^\top$ as follows:

$$\mathbf{y}_i = \text{MLP}(\tilde{\mu}_i; \mathbf{W}). \quad (7)$$

4.2 Learning Procedure

In RKM-Net learning, we learns parameters $\Theta = \{\mathbf{V}, \mathbf{W}\}$, where $\mathbf{V} = \{\mathbf{v}_s\}_{s\in\mathcal{V}}$ is a set of latent vectors, and $\mathbf{W}$ is the weight parameters for MLP.

First, we define a loss function for RKM-Nets. In general, it

[*3] Although a typical multiplicity function is defined as $m_i : \mathcal{A}_i \to \mathbb{N}_+$, we expand the definition to include the real-valued importance of features such as TF-IDF.

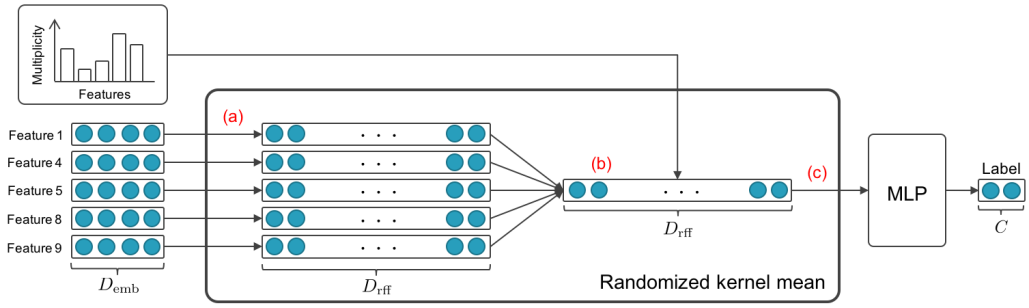

Fig. 1 Model architecture of a randomized kernel mean network for classification. The inputs are a set of features associated with each sample and their multiplicity e.g., feature frequency or TF-IDF. Each feature is converted to its corresponding D_{emb}-dimensional latent vector. **(a)** For each latent vector, a D_{rff}-dimensional vector is obtained by applying a random Fourier feature algorithm, as described in Section 3. **(b)** This computes the mean of the obtained vectors weighted by the multiplicity of their features, which we call the *randomized kernel mean*. **(c)** The randomized kernel mean is used as input in MLP. Finally, the MLP outputs label confidence with C units.

is possible to use any loss functions such as hinge loss and cross entropy loss. In this study, we consider an ℓ_1 hinge loss used for one-of-many classification. The hinge loss of each sample i is defined as

$$h_i(\Theta) = \sum_{c=1}^{C} \max\{0, 1 - \delta(t_i = c) y_{ic}\}, \tag{8}$$

where, function δ is an indicator function that returns 1 if $t_i = c$ and 0 otherwise, and y_{ic} is the RKM-Net output for label c of the ith sample. Intuitively, the loss becomes small when the value of y_{ic} with the correct label c is large and that of $y_{ic'}$ with the other label $c' \neq c$ is small.

The parameters Θ can be learned by minimizing the loss (8) via a standard backpropagation algorithm with stochastic gradient descent (SGD). With SGD, we iteratively continue to 1) compute the gradient with respect to the parameters via backpropagation in randomly selected samples without overlaps, i.e., mini-batch, and to 2) update the parameters based on the gradient descent method. In particular, we first define the loss for mini-batch $\mathcal{B} \subseteq \{1, 2, \cdots, N_{\text{tr}}\}$ as follows:

$$\mathcal{L}_{\mathcal{B}}(\Theta) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} h_i(\Theta). \tag{9}$$

Then, we compute the gradients of the loss with respect to the parameters $\mathbf{W}, \mathbf{V}$. The gradient of $\mathbf{W}$ is as follows:

$$\frac{\partial \mathcal{L}_{\mathcal{B}}(\Theta)}{\partial \mathbf{W}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{\partial h_i(\Theta)}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i}{\partial \mathbf{W}}. \tag{10}$$

Because the gradient of the hinge loss $\frac{\partial h_i(\Theta)}{\partial \mathbf{y}_i}$ and the gradient of MLP $\frac{\partial \mathbf{y}_i}{\partial \mathbf{W}}$ were already derived in previous studies, e.g. [23], we do not describe it here in detail. For each feature $s \in \mathcal{V}$, the gradient with respect to $\mathbf{v}_s$ can be obtained as follows:

$$\frac{\partial \mathcal{L}_{\mathcal{B}}(\Theta)}{\partial \mathbf{v}_s} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{\partial h_i(\Theta)}{\partial \mathbf{y}_i} \frac{\partial \mathbf{y}_i}{\partial \tilde{\mu}_i} \frac{\partial \tilde{\mu}_i}{\partial \mathbf{z}_s} \frac{\partial \mathbf{z}_s}{\partial \mathbf{v}_s}. \tag{11}$$

Here, the newly derived gradients for RKM-Net are $\frac{\partial \tilde{\mu}_i}{\partial \mathbf{z}_s}$ and $\frac{\partial \mathbf{z}_s}{\partial \mathbf{v}_s}$, which are formed as $D_{\text{rff}} \times D_{\text{rff}}$ and $D_{\text{rff}} \times D_{\text{emb}}$ matrices, respectively. These gradients are given by

$$\frac{\partial \tilde{\mu}_i}{\partial \mathbf{z}_s} = \frac{m_i(s)}{\#x_i} \mathbf{I}_{D_{\text{rff}}}, \quad \left(\frac{\partial \mathbf{z}_s}{\partial \mathbf{v}_s}\right)_{lk} = -\sqrt{2} \sin\left(\mathbf{r}_l^\top \mathbf{v}_s + s_l\right) r_{lk}. \tag{12}$$

Table 1 Dataset specifications. Here, N_{tr}, N_{te} and C are the number of training samples, the number of test samples and the number of classes, respectively. Note that the vocabulary size $|\mathcal{V}|$ of 20Ng and Cade12 is limited to 10,000 by trimming the low-frequency features.

	N_{tr}	N_{te}	$\lvert\mathcal{V}\rvert$	C
R8	5,485	2,189	14,575	8
R52	6,532	2,568	16,145	52
20Ng	11,239	7,528	10,000	20
Cade12	27,322	13,661	10,000	12
WebKb	2,803	1,396	7,287	4

Table 2 Accuracy of BoW document classification.

	R8	R52	20Ng	Cade12	WebKb
Naive Bayes	0.960	0.869	0.810	0.573	0.835
Linear SVM	0.970	0.938	**0.828**	0.528	0.858
Poly SVM	0.972	0.917	0.815	0.577	**0.913**
DNN	0.974	0.942	0.818	0.572	0.897
Mean Pooling	0.963	0.912	0.690	0.504	0.908
RKM-Net	**0.978**	**0.949**	0.817	**0.579**	0.897

5. Experiments

We experiment on BoW document classification with five publicly open datasets [2], namely, R8, R52, 20Ng, Cade12, and WebKb. Each dataset comprises documents with single labels, and their training/test splitting was already performed. The datasets' details are available on the author's webpage[*4]. Their specifications are shown in Table 1.

RKM-Net is implemented in Chainer [24][*5], a deep neural network framework written in Python. For the implementation of RKM-Net, we use a MLP with a hidden layer. For the hidden layer, we use a rectified linear unit (ReLU) as an activation function and Dropout for regularization. The hyperparameters of the RKM-Net include the band-width parameter γ for the RFF algorithm, the dimensionality of the latent vector D_{emb}, the dimensionality of the random features D_{rff} and the Dropout rate r. We search the optimal hyperparameters by grid search using the ranges $\gamma \in \{2^{-3}, 2^{-1}, \cdots, 2^5\}$, $D_{\text{emb}} \in \{100, 200, 300\}$, $D_{\text{rff}} \in \{500, 1000, 1500\}$ and $r \in \{0.1, 0.3, 0.5\}$. We initialize the latent vectors for features and weights for MLP by samples generated from Gaussian $\mathcal{N}(0, 1)$, and determine the learning rate of

[*4] http://ana.cachopo.org/datasets-for-single-label-text-categorization
[*5] http://chainer.org/

SGD using Adam [9].

We compare the RKM-Net with Naive Bayes, Linear SVM, Poly(nomial) SVM, DNN and mean pooling. For Naive Bayes and Linear SVM, we simply use the results described in [2]. As for Poly SVM, we use the SVM implementation in scikit-learn[*6]. The hyperparameters for Poly SVM are the degree d of the polynomial kernels and the cost parameter C of the SVM. We search the optimal hyper-parameters by grid search using the ranges $d \in \{2, 3, 4, 5\}$ and $C \in \{10^{-5}, 10^{-4}, \cdots, 10^{4}\}$. With DNN, we use a MLP with two hidden layers; the ReLU activation function is used for both hidden layers and Dropout is adopted only for the second layer. The hyperparameters for DNN are the number of units for the hidden layers U and the Dropout rate r, which are grid-searched within the ranges $U \in \{200, 400, \cdots, 1000\}$ and $r \in \{0.1, 0.3, 0.5\}$. Mean pooling is a neural network that passes the mean of latent vectors for features associated with each sample to a MLP with a hidden layer. Thus, we can determine the optimum representation by comparing the mean vector in mean pooling with the randomized kernel mean in the RKM-Net. The hyperparameters for mean pooling are the number of units in the hidden layers U and the Dropout rate r, which are searched as with DNN. We initialize all the parameters in DNN and mean pooling by samples generated from Gaussian $\mathcal{N}(0, 1)$, and determine the learning rate of SGD using Adam.

Table 2 shows the BoW document classification accuracy for the five datasets. RKM-Net achieves the best accuracy for the three datasets: R8, R52, and Cade12. For 20Ng and WebKb, the best accuracy is achieved by the linear kernel and the polynomial kernel SVMs, respectively. Among deep learning models, RKM-Net outperforms DNN and mean pooling except for 20Ng and WebKb datasets. Note that the difference between RKM-Net and mean pooling is that RKM-Net uses the randomized kernel mean of the latent vectors for each sample while mean pooling uses the mean of them. As a result, the difference is essential to improve the accuracy for the four datasets.

In Figure 2, we illustrate the BoW count vectors and the RKM-Net randomized kernel mean by applying Isomap, which is a nonlinear dimension reduction method. With BoW count vectors, samples with different classes are overlapped. Obviously, the result implies that classifying the samples correctly without representation learning is difficult. Conversely, with RKM-Net, the samples with the same label are closer distributed, and samples with different labels are placed at greater distances from each other as the RKM-Net training proceeds from epoch 1 to epoch 6. This result indicates that RKM-Net can learn the representation of samples valuable for classification.

In Figure 3, we show the densities of the learned latent vectors for each class on the WebKb dataset. We can regard the density of each class as the density of the latent vectors for a sample associated with the class. As shown in this figure, each density has a single or multiple peaks at different location. Because RKM-Net classify the samples by capturing the shape of the density, the result indicates that the latent vector are learned so as to classify the samples correctly.

[*6] http://scikit-learn.org/stable/

6. Conclusion

Deep learning has attracted attention as it is useful for various tasks with different types of data such as image recognition and machine translation. However, there are no deep learning models specific for BoW data, despite BoW data appears in wide variety of areas such as natural language processing, bioinformatics and data mining. In this study, we have proposed a deep learning model for BoW data, which we call it Randomized Kernel Mean Networks (RKM-Nets). To represent BoW data effectively, RKM-Nets assume that each feature is associated with a latent vector, and represent each sample as a distribution of latent vectors of features contained in the sample. Then, the distribution is represented as a randomized kernel mean vector that approximately holds the moment information about the distribution by combining the kernel mean embeddings and random Fourier feature algorithm. By using the randomized kernel mean vectors as input of existing neural networks, we can employ deep learning for various types of machine learning problems for BoW data. In this study, we have performed BoW data classification by using a simple multi-layer perceptron in our framework. In future works, we will further confirm the effectiveness of the framework of RKM-Nets by applying this for some applications such as multimodal learning.

References

[1] Baltrunas, L., Makcinskas, T. and Ricci, F.: Group Recommendations with Rank Aggregation and Collaborative Filtering, *RecSys* (2010).
[2] Cardoso-Cachopo, A.: Improving Methods for Single-label Text Categorization, PhD Thesis (2007).
[3] Dziugaite, G. K., Roy, D. M. and Ghahramani, Z.: Training Generative Neural Networks via Maximum Mean Discrepancy Optimization, *arXiv:1505.03906* (2015).
[4] Gretton, A., Fukumizu, K., Teo, C., Song, L., Schölkopf, B. and Smola, A.: A Kernel Statistical Test of Independence, *Advances in Neural Information Processing Systems* (2008).
[5] Guo, X., Lee, H., Wang, X. and Lewis, R. L.: Deep Learning for Real-time Atari Game Play Using Offline Monte Carlo Tree Search Planning, *Advances in Neural Information Processing Systems* (2014).
[6] He, K., Zhang, X., Ren, S. and Sun, J.: Deep Residual Learning for Image Recognition, *Arxiv.Org*, Vol. 7, No. 3, pp. 171–180 (online), DOI: 10.3389/fpsyg.2013.00124 (2015).
[7] Johnson, R. and Zhang, T.: Effective Use of Word Order for Text Categorization with Convolutional Neural Networks, *NAACL HLT* (2015).
[8] Kim, Y.: Convolutional Neural Networks for Sentence Classification, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1746–1751 (2014).
[9] Kingma, D. and Ba, J.: Adam: A Method for Stochastic Optimization, *Proceedings of the 3rd International Conference for Learning Representations* (2015).
[10] Lai, S., Xu, L., Liu, K. and Zhao, J.: Recurrent Convolutional Neural Networks for Text Classification, *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 2267–2273 (2015).
[11] Li, Y., Swersky, K. and Zemel, R.: Generative Moment Matching Networks, *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1718–1727 (2015).
[12] Muandet, K., Fukumizu, K., Dinuzzo, F. and Schölkopf, B.: Learning from Distributions via Support Measure Machines, *Advances in Neural Information Processing Systems* (2012).
[13] Oliva, J. B., Sutherland, D. J. and Schneider, J.: Deep Mean

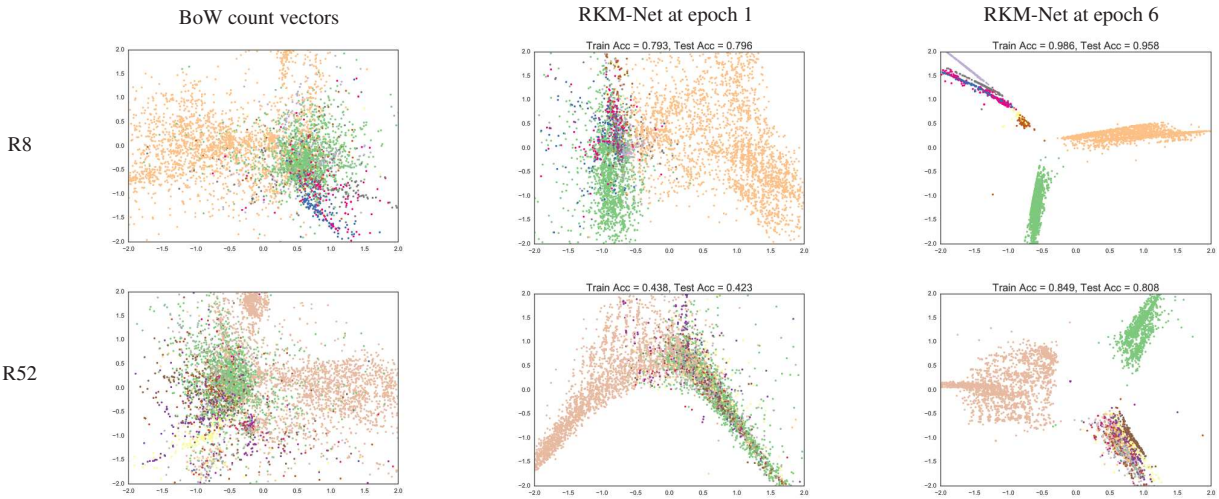

Fig. 2 Visualization of BoW count vectors and randomized kernel means of RKM-Nets. Each dot indicates a sample (document), and its color denotes its corresponding label. The rows indicate datasets: R8 and R52 datasets. The first column shows the results using BoW vectors directly. The second and third columns show the results using random kernel means at epochs 1 and 6, respectively.

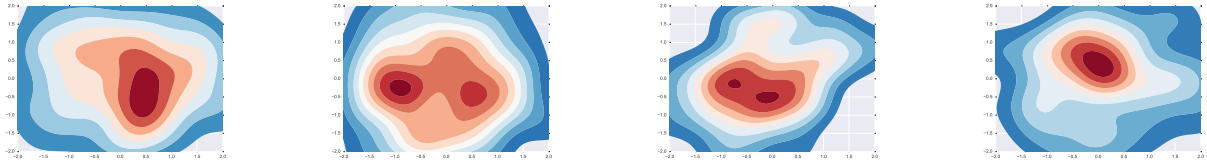

Fig. 3 Visualization of the densities of the learned latent vectors in RKM-Net on the WebKb dataset. Each figure corresponds to one class. After performing dimension reduction of the latent vectors via Isomap, the density for each class is obtained by using the latent vectors of features characteristically appearing in the samples with the class.

Maps, *ArXiv* (2015).

[14] Rahimi, A. and Recht, B.: Random Features for Large-Scale Kernel Machines, *Advances in Neural Information Processing Systems* (2008).

[15] Rudin, W.: *Fourier Analysis on Groups* (1962).

[16] Shivashankar, S., Srivathsan, S., Ravindran, B. and Tendulkar, A. V.: Multi-view Methods for Protein Structure Comparison Using Latent Dirichlet Allocation, *Bioinformatics*, Vol. 27, No. 13, pp. 61–68 (online), DOI: 10.1093/bioinformatics/btr249 (2011).

[17] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D.: Mastering The Game of Go with Deep Neural Networks and Tree Search, *Nature*, Vol. 529, No. 7587, pp. 484–489 (online), DOI: 10.1038/nature16961 (2016).

[18] Smola, A., Gretton, A., Song, L. and Schölkopf, B.: A Hilbert Space Embedding for Distributions, *Algorithmic Learning Theory* (2007).

[19] Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B. and Lanckriet, G. R. G.: Hilbert Space Embeddings and Metrics on Probability Measures, *The Journal of Machine Learning Research*, Vol. 11, pp. 1517–1561 (2010).

[20] Sun, A., Bhowmick, S. S., Nguyen, K. N. and Bai, G.: Tag-Based Social Image Retrieval: An Empirical Evaluation, *Journal of the American Society for Information Science and Technology*, Vol. 62, No. 12, pp. 2364–2381 (online), DOI: 10.1002/asi (2011).

[21] Sutskever, I., Vinyals, O. and Le, Q. V.: Sequence to Sequence Learning with Neural Networks, *Advances in Neural Information Processing Systems*, pp. 3104–3112 (2014).

[22] Tang, D., Qin, B. and Liu, T.: Document Modeling with Gated Recurrent Neural Network for Sentiment Classification, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, No. September, pp. 1422–1432 (online), available from ⟨http://aclweb.org/anthology/D15-1167⟩ (2015).

[23] Tang, Y.: Deep Learning using Linear Support Vector Machines, *International Conference on Machine Learning 2013: Challenges in Representation Learning Workshop* (2013).

[24] Tokui, S., Oono, K., Hido, S. and Clayton, J.: Chainer: a Next-Generation Open Source Framework for Deep Learning, *Proceedings of Workshop on Machine Learning Systems(LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems* (2015).

[25] Yoshikawa, Y., Iwata, T. and Sawada, H.: Latent Support Measure Machines for Bag-of-Words Data Classification, *Advances in Neural Information Processing Systems* (2014).

[26] Yoshikawa, Y., Iwata, T. and Sawada, H.: Non-linear Regression for Bag-of-Words Data via Gaussian Process Latent Variable Set Model, *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (2015).

[27] Yoshikawa, Y., Iwata, T., Sawada, H. and Yamada, T.: Cross-Domain Matching via Kernel Embeddings of Latent Distributions, *Advances in Neural Information Processing Systems* (2015).