A General Representation Learning Framework with Generalization Performance Guarantees

Junbiao Cui [1] **Jianqing Liang** [1] **Qin Yue** [1] **Jiye Liang** [1]

Abstract

The generalization performance of machine learning methods depends heavily on the quality of data representation. However, existing researches rarely consider representation learning from the perspective of generalization error. In this paper, we prove that generalization error of representation learning function can be estimated effectively by solving two convex optimization problems. Based on it, we propose a general representation learning framework. And then, we apply the proposed framework to two most commonly used nonlinear mapping methods, i.e., kernel based method and deep neural network (DNN), and thus design a kernel selection method and a DNN boosting framework, correspondingly. Finally, extensive experiments verify the effectiveness of the proposed methods.

1. Introduction

Machine learning, especially supervised learning, has achieved significant success in many fields, such as computer vision (Russakovsky et al., 2015; He et al., 2016; Carion et al., 2020; He et al., 2022), speech recognition (Sainath et al., 2013; Gulati et al., 2020; Chiu et al., 2022), and natural language processing (Brown et al., 2020; Qiu et al., 2020; Cuadros et al., 2022), etc. The nature of machine learning is to learn the law from empirical data, which is usually a function from input space $\mathcal{X}$ to output space $\mathcal{Y}$, i.e., $f : \mathcal{X} \to \mathcal{Y}$.

It is well known that the data representation has a huge effect on the performance of machine learning methods (Goodfellow et al., 2016). Better data representations can make learning tasks easier to solve. In most cases, the function f is nonlinear, thus we need to apply a nonlinear mapping φ to input space $\mathcal{X}$. The process of learning φ from data is called representation learning.

In machine learning, two most commonly used nonlinear mapping methods are kernel based method and DNN.

Kernel based method utilizes kernel mapping φ_{kernel} to map original data into a latent space $\mathcal{X}'$, and then completes learning in latent space. In general, it is not necessary to give explicit expression of kernel mapping φ_{kernel}, making the kernel based method highly flexible and universal. Benefiting from it, a large number of kernel based methods have been proposed, such as kernelized principal component analysis (Schölkopf et al., 1998), kernelized support vector machine (Cortes & Vapnik, 1995), and kernelized k-means (Zhang & Rudnicky, 2002), etc. Among them, the selection of kernel directly determines the generalization performance of kernel based method. So far, many kernels have been proposed, such as Gaussian kernel, polynomial kernel, and Laplacian kernel, etc. However, selecting a proper kernel for a learning task is very challenging, because it is difficult to estimate the generalization performance of the corresponding latent data representation given a kernel.

DNN (LeCun et al., 2015) maps original data into a latent space $\mathcal{X}'$ through a multi-layer neural network φ_{DNN}, and then adds a liner mapping at the end of φ_{DNN}. In DNN, the network architecture and loss are major impact factors of the generalization performance.

For a learning task, designing a good network architecture usually needs domain knowledge. For example, CNN (LeCun et al., 1989; Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; He et al., 2016), ViT (Dosovitskiy et al., 2021), and MAE (He et al., 2022) are designed for computer vision tasks, while RNN (Rumelhart et al., 1986), LSTM (Hochreiter & Schmidhuber, 1997), Seq2Seq (Sutskever et al., 2014), BERT (Devlin et al., 2019), and GPT-3 (Brown et al., 2020) are designed for natural language processing tasks. How to design good network architectures for specific tasks is beyond the scope of this paper.

So far, a lot of losses have been proposed to train DNN, which can be roughly divided into two categories, super-

[1]Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 030006, Shanxi, China. Correspondence to: Jiye Liang <ljy@sxu.edu.cn>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

vised loss and unsupervised regularization term loss.

Classical supervised losses includes cross entropy loss and mean squared error, etc. These losses are calculated based on labeled data and are surrogate functions of empirical error. When labeled data is insufficient, their ability to estimate generalization error will be limited.

p-norm family (Hanson & Pratt, 1988; Loshchilov & Hutter, 2019) is popular unsupervised regularization term loss. And these losses are designed based on domain knowledge or experience. Therefore, they are lack of universality. In addition, it is difficult to give theoretical guarantee whether these losses are beneficial to reduce generalization error of DNN.

In summary, in order to learn a good nonlinear mapping φ (select a proper kernel for kernel based method and design an universal loss for DNN), it is indispensable to design a criterion to estimate generalization error of data representation in latent space $\mathcal{X}'$ and guide representation learning. This paper will achieve this goal. Specifically,

- A VC dimension based criterion is proposed to measure generalization error of representation learning function. It is proved that the criterion can be calculated effectively by solving two convex optimization problems. And then algorithms are designed to solve the corresponding optimization problems efficiently.
- Based on the criterion, we propose a general representation learning framework, which aims to minimize generalization error of representation learning function.
- Based on the framework, we design a kernel selection method for kernel based method and a boosting framework for DNN.
- A toy example demonstrates the effectiveness of the proposed criterion for measuring generalization error of data representation. And systematic experiments verify the effectiveness of the proposed framework for kernel selection and DNN boosting.

2. Notations and Preliminaries

2.1. Problem Formalization

Let $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} = \{1, 2, \cdots, K\}$, $K \geq 2$ and $f_{\mathrm{c}} : \mathcal{X} \to \mathcal{Y}$ be input (feature) space, output (class label) space and unknown target function, respectively. Let $\varphi : \mathcal{X} \to \mathcal{X}'$, $\mathcal{X}' \subseteq \mathbb{R}^{d'}$, be representation learning function.

For sake of discussion, this paper takes binary classification problem as an example to build the theory and method. By using "one-versus-one" or "one-versus-rest" strategy, the multiple classification problem can be converted into a series of binary ones, so that the proposed methods can work directly. Moreover, recent study shows that arbitrary classification problem can be equivalently converted into a binary classification problem (Cui & Liang, 2022).

Given a binary classification problem, let $\mathcal{X} = \mathcal{X}_+ \bigcup \mathcal{X}_-$ and $\mathcal{Y} = \{+1, -1\}$ be input space and output space, respectively. $\mathcal{X}_+ \subseteq \mathbb{R}^d$ and $\mathcal{X}_- \subseteq \mathbb{R}^d$ are positive samples space and negative samples space, respectively. Let $X_+ = \{\mathbf{x}_i | \mathbf{x}_i \in \mathcal{X}_+, i = 1, 2, \cdots, n_+\}$ and $X_- = \{\mathbf{x}_j | \mathbf{x}_j \in \mathcal{X}_-, j = n_+ + 1, n_+ + 2, \cdots, n_+ + n_-\}$ be the positive and negative training samples, respectively. And let $X = X_+ \bigcup X_-$ and $n = n_+ + n_-$ be set of all training samples and the number of all training samples, respectively.

At the same time, main notations used in this paper are listed in **Table** 1.

Table 1. Definition of main notations.

NOTATION	DEFINITION
$\varphi(\mathcal{X}_+), \varphi(\mathcal{X}_-)$	$\{\varphi(\mathbf{x}) \|\mathbf{x} \in \mathcal{X}_+\}, \{\varphi(\mathbf{x}) \|\mathbf{x} \in \mathcal{X}_-\}$
$\varphi(\mathcal{X})$	$\varphi(\mathcal{X}_+) \bigcup \varphi(\mathcal{X}_-)$
$\varphi(X_+), \varphi(X_-)$	$\{\varphi(\mathbf{x}) \|\mathbf{x} \in X_+\}, \{\varphi(\mathbf{x}) \|\mathbf{x} \in X_-\}$
$\varphi(X)$	$\varphi(X_+) \bigcup \varphi(X_-)$
$\varphi(\mathbf{X}_+)$	$\left(\varphi(\mathbf{x}_1), \cdots, \varphi\left(\mathbf{x}_{n_+}\right)\right) \in \mathbb{R}^{d' \times n_+}$
$\varphi(\mathbf{X}_-)$	$\left(\varphi\left(\mathbf{x}_{n_+ + 1}\right), \cdots, \varphi(\mathbf{x}_n)\right) \in \mathbb{R}^{d' \times n_-}$
$\varphi(\mathbf{X})$	$(\varphi(\mathbf{X}_+), \varphi(\mathbf{X}_-)) \in \mathbb{R}^{d' \times n}$
$\triangle^m$	$\{\mathbf{v} \|\mathbf{v} \in \mathbb{R}^m, \mathbf{v} \geq \mathbf{0}, \sum_{i=1}^m v_i = 1\}$
σ	$\sigma : \mathbb{R}^m \to \triangle^m$, $\forall \mathbf{v} \in \mathbb{R}^m$, $\sigma(\mathbf{v}) = \frac{1}{z}(\exp(v_1), \cdots, \exp(v_m))^T$, $z = \sum_{i=1}^m \exp(v_i)$

2.2. Preliminaries of VC Dimension

Definition 2.1 (Chapter 3.6 of (Vapnik, 1999))**.** Let $\mathcal{H}$ be a set of real functions. The VC dimension of $\mathcal{H}$, denoted as $d_{VC}(\mathcal{H})$, is defined as the maximum m of vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$ that can be separated into two classes in all 2^m possible ways using functions of the set $\mathcal{I}(\mathcal{H})$, where $\mathcal{I}(\mathcal{H}) = \{g | g(\mathbf{x}) = \mathbb{I}(h(\mathbf{x}) > b), h \in \mathcal{H}, b \in \mathbb{R}\}$ is the set of indicator functions corresponding to $\mathcal{H}$, and $\mathbb{I}(\cdot)$ is 1, if $\cdot$ is true, and is 0, if $\cdot$ is false.

Definition 2.2 (Chapter 5.4.2 of (Vapnik, 1999))**.** Given a hyperplane $ph(\boldsymbol{\omega}, b)$, $\|\boldsymbol{\omega}\|_2 = 1$, $b \in \mathbb{R}$. The $ph(\boldsymbol{\omega}, b)$ is called as M-margin separating hyperplane if it classifiers vectors $\mathbf{x}$ as follows

$$y = \begin{cases} +1, & if\ \boldsymbol{\omega}^T \mathbf{x} + b \geq M \\ -1, & if\ \boldsymbol{\omega}^T \mathbf{x} + b \leq -M \end{cases}.$$

Theorem 2.3 (Theorem 5.1 of (Vapnik, 1999))**.** *Let vectors $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ belong to a sphere of radius R. Then the set of*

M-margin separating hyperplanes has VC dimension d_{VC} bounded by the inequality

$$d_{VC} \leq B_1(d, R, M) = min\left(\left[\frac{R^2}{M^2}\right], d\right) + 1.$$

Theorem 2.4 (Corollary in Chapter 5.4 of (Vapnik, 1999))**.** *With probability $1-\eta$ one can assert that the probability that a test sample will not be separated correctly by the M-margin hyperplane has the bound $P_{err} \leq \frac{n_{err}}{n} + B_2(n, n_{err}, \eta, d_{VC})$, where $B_2(n, n_{err}, \eta, d_{VC}) = \frac{\varepsilon}{2}\left(1+\sqrt{1+\frac{4n_{err}}{n\mathcal{E}}}\right)$, $\mathcal{E} = 4\frac{d_{VC}\left(\ln\frac{2n}{d_{VC}}+1\right)-\ln\frac{\eta}{4}}{n}$, n is the number of training samples, n_{err} is the number of training samples that are not separated correctly by this M-margin hyperplane, and d_{VC} is the VC dimension in **Theorem** 2.3.*

3. VC Dimension based Representation Learning Framework

3.1. Formalize Generalization Error of Representing Learning

Given a learning task, the learning process is to obtain a function $\hat{f}: \mathcal{X} \to \mathcal{Y}$ to approximate the unknown target function f_c by using training data. Generally, function $\hat{f}$ can be written as the composition of two functions, i.e., $\hat{f} = h \circ \varphi$, where $\varphi: \mathcal{X} \to \mathcal{X}'$ is representation learning function and h is classification function. The learning process can be written as follows.

Representation learning can be formalized as follows

$$\varphi^* = \underset{\varphi\in\Psi}{arg\,min}\ P_{err}(\mathcal{H}(\varphi(\mathcal{X}))) \quad , \tag{1}$$

where Ψ is the set of candidate representation learning functions, $P_{err}(\mathcal{H}(\varphi(\mathcal{X})))$ is generalization error of $\mathcal{H}(\varphi(\mathcal{X}))$, and $\mathcal{H}(\varphi(\mathcal{X})) = \{h | h \text{ is a hyperplne on space } \varphi(\mathcal{X})\}$ is hypothesis space of classifiers. Given $\mathcal{X}$, $P_{err}(\mathcal{H}(\varphi(\mathcal{X})))$ is uniquely determined by representation learning function $\varphi(\cdot)$, so it is actually generalization error of representation learning.

Classifier learning can be formalized as follows

$$h^* = \underset{h\in\mathcal{H}(\varphi^*(\mathcal{X}))}{arg\,min}\ \mathcal{L}(h, \varphi^*(X)) + \gamma_{\mathrm{R}}\mathcal{R}(h) \quad , \tag{2}$$

where $\mathcal{L}$ is the loss function that measures how well the h fits the training samples X, $\mathcal{R}$ is the regularization term, and $\gamma_{\mathrm{R}} > 0$ is the trade-off parameter.

The above two components can be executed sequentially (an example is given in **Section** 4). They also can be integrated into an unified model for end-to-end learning (an example is given in **Section** 5).

3.2. A Upper Bound of $P_{err}(\mathcal{H}(\varphi(\mathcal{X})))$

According to **Theorem** 2.4, we can reduce generalization error P_{err} in formula (1) by minimizing its upper bound $\frac{n_{err}}{n} + B_2(n, n_{err}, \eta, d_{VC}(\mathcal{H}(\varphi(\mathcal{X}))))$, so formula (1) can be rewritten as follows

$$\begin{array}{l} \underset{\varphi\in\Psi}{min}\ B_2(n, n_{err}, \eta, d_{VC}(\mathcal{H}(\varphi(\mathcal{X})))) \\ s.t.\ \frac{n_{err}}{n} \leq \varepsilon \end{array} \quad , \tag{3}$$

where $\frac{n_{err}}{n}$ is the empirical error on training set, $\varepsilon > 0$, $\varepsilon \approx 0$, and $\frac{n_{err}}{n} \leq \varepsilon$ is a constraint easy to satisfy (see **Remark** A.5 for details). And $d_{VC}(\cdot)$ is VC dimension of set of functions $\cdot$.

By removing variables unrelated to φ, formula (3) can be written as follows

$$\underset{\varphi\in\Psi}{min}\ d_{VC}(\mathcal{H}(\varphi(\mathcal{X})))\ \ s.t.\ \ \frac{n_{err}}{n} \leq \varepsilon \quad . \tag{4}$$

It's hard to directly minimize the VC dimension (see **Definition** 2.1). According to **Theorem** 2.3, we can indirectly control the VC dimension by minimizing its upper bound, so formula (4) can be rewritten as follows

$$\begin{array}{l} \underset{\varphi\in\Psi}{min}\ min\left(\left[\frac{R^2(\varphi(\mathcal{X}))}{M^2(\varphi(\mathcal{X}_+),\varphi(\mathcal{X}_-))}\right], d(\varphi(\mathcal{X}))\right) + 1 \\ s.t. \begin{cases} \frac{n_{err}}{n} \leq \varepsilon \\ M(\varphi(\mathcal{X}_+), \varphi(\mathcal{X}_-)) = \underset{h\in\mathcal{H}(\varphi(\mathcal{X}))}{sup}\ g_{\mathrm{m}}(h) \end{cases} \end{array} \quad , \tag{5}$$

where $R(\varphi(\mathcal{X}))$ and $d(\varphi(\mathcal{X}))$ are radius and dimension of latent space $\varphi(\mathcal{X})$, respectively. $g_{\mathrm{m}}(h)$ is the supremum of margin of h on $\varphi(\mathcal{X})$ given target function f_c (see **Definition** 2.2).

In general, $d(\varphi(\mathcal{X})) > \left[\frac{R^2(\varphi(\mathcal{X}))}{M^2(\varphi(\mathcal{X}_+),\varphi(\mathcal{X}_-))}\right]$, so formula (5) can be rewritten as follows

$$\begin{array}{l} \underset{\varphi\in\Psi}{min}\ \frac{R^2(\varphi(\mathcal{X}))}{M^2(\varphi(\mathcal{X}_+),\varphi(\mathcal{X}_+))} \\ s.t. \begin{cases} \frac{n_{err}}{n} \leq \varepsilon \\ M(\varphi(\mathcal{X}_+), \varphi(\mathcal{X}_+)) = \underset{h\in\mathcal{H}(\varphi(\mathcal{X}))}{sup}\ g_{\mathrm{m}}(h) \end{cases} \end{array} . \tag{6}$$

Formula (6) is not executable, because the distribution of input space $\mathcal{X}$ is unknown. Fortunately, training set $X = X_+ \bigcup X_-$ that is sampled from $\mathcal{X}$ is available. Therefore, we can use X to estimate the distribution of $\mathcal{X}$, which results in the following optimization problem

$$\begin{array}{l} \underset{\varphi\in\Psi}{min}\ \frac{R^2(\varphi(X))}{M^2(\varphi(X_+),\varphi(X_-))} \\ s.t. \begin{cases} \frac{n_{err}}{n} \leq \varepsilon \\ M(\varphi(X_+), \varphi(X_-)) = \underset{h\in\mathcal{H}(\varphi(X))}{sup}\ g_{\mathrm{m}}(h) \end{cases} \end{array} . \tag{7}$$

Formula (7) is still not executable, because margin $M(\varphi(X_+), \varphi(X_-))$ and radius $R(\varphi(X))$ are both intractable. The next section will address these two problems.

3.3. Making Margin and Radius Executable

For sake of discussion, this section takes input space $\mathcal{X}$ as an example to solve the margin and radius problems. The results can be directly applied to latent space $\varphi(\mathcal{X})$.

3.3.1. Margin

In formula (7), constraint

$$M\left(X_+, X_-\right) = \sup_{h \in \mathcal{H}(X)} g_{\mathrm{m}}\left(h\right) \tag{8}$$

is intractable. In order to solve it effectively, a lower bound of $M\left(X_+, X_-\right)$ is constructed by the following theorem.

Theorem 3.1. *The optimization problem (8) can be bounded by the following convex optimization problem. See **Appendix** A.1 for proof.*

$$M^2\left(X_+, X_-\right) \geq \frac{1}{4} \min_{\boldsymbol{\alpha} \in \triangle^{n_+},\, \boldsymbol{\beta} \in \triangle^{n_-}} \|\boldsymbol{X}_+ \boldsymbol{\alpha} - \boldsymbol{X}_- \boldsymbol{\beta}\|_2^2$$

3.3.2. Radius

In formula (7), $R\left(X\right)$ is radius of set X. To calculate it effectively, we first formalize $R\left(X\right)$ by **Definition** A.7 in **Appendix** A.2 and then prove that $R\left(X\right)$ can be calculated by solving a convex optimization.

Theorem 3.2. *Given a set $X \subset \mathbb{R}^d$, the squared radius of X can be computed by the following convex optimization problem. See **Appendix** A.2 for proof.*

$$R^2(X) = \inf_{\boldsymbol{\theta} \in \triangle^{|X|}} \sup_{\boldsymbol{x} \in X} \|\boldsymbol{x} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2$$

3.4. Final Model and Solving

By substituting the conclusions of **Theorems** 3.1 and 3.2 into optimization problem (7), we have

$$\begin{aligned} &\min_{\varphi \in \Psi} \; f(\varphi) = 4\frac{g_1(\varphi)}{g_2(\varphi)} \\ &s.t. \begin{cases} g_1\left(\varphi\right) = \min\limits_{\boldsymbol{\theta}} \; \max\limits_{\mathbf{x}_i \in X} \; \|\varphi\left(\mathbf{x}_i\right) - \varphi\left(\mathbf{X}\right)\boldsymbol{\theta}\|_2^2 \\ \qquad\quad s.t. \;\; \boldsymbol{\theta} \in \triangle^n \\ g_2\left(\varphi\right) = \min\limits_{\boldsymbol{\alpha},\, \boldsymbol{\beta}} \; \|\varphi\left(\mathbf{X}_+\right)\boldsymbol{\alpha} - \varphi\left(\mathbf{X}_-\right)\boldsymbol{\beta}\|_2^2 \\ \qquad\quad s.t. \;\; \boldsymbol{\alpha} \in \triangle^{n_+},\; \boldsymbol{\beta} \in \triangle^{n_-} \end{cases} \end{aligned} . \tag{9}$$

The geometric meaning of formula (9) is clear. $g_1\left(\varphi\right)$ is radius of training samples in latent space (**Definition** A.7 in **Appendix** A.2) and $g_2\left(\varphi\right)$ is the distance between the convex hull of positive training samples and the convex hull of negative training samples in latent space (**Definition** A.2 in **Appendix** A.1).

In formula (9), $g_1\left(\varphi\right)$ and $g_2\left(\varphi\right)$ are both convex quadratic optimization problems and there are many conventional solving methods (Boyd & Vandenberghe, 2004). To solve them more efficiently, the constraint $\triangle^m$ is expanded by softmax activation function σ, i.e.,

$$\begin{aligned} &\min_{\varphi \in \Psi} \; f(\varphi) = 4\frac{g_1(\varphi)}{g_2(\varphi)} \\ &s.t. \begin{cases} g_1\left(\varphi\right) = \min\limits_{\mathbf{u}} \; \max\limits_{\mathbf{x}_i \in X} \; \|\varphi\left(\mathbf{x}_i\right) - \varphi\left(\mathbf{X}\right)\sigma\left(\mathbf{u}\right)\|_2^2 \\ g_2\left(\varphi\right) = \min\limits_{\mathbf{v},\mathbf{w}} \; \|\varphi\left(\mathbf{X}_+\right)\sigma\left(\mathbf{v}\right) - \varphi\left(\mathbf{X}_-\right)\sigma\left(\mathbf{w}\right)\|_2^2 \end{cases} \end{aligned} , \tag{10}$$

where $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{v} \in \mathbb{R}^{n_+}$, $\mathbf{w} \in \mathbb{R}^{n_-}$. The optimization problems $g_1\left(\varphi\right)$ and $g_2\left(\varphi\right)$ can be solved effectively and efficiently by using the accelerated first-order algorithm, e.g., Adam (Kingma & Ba, 2015) and high-performance computing hardware, e.g., GPU. See **Appendix** B.1 for details.

4. VC Dimension based Kernel Selection

In this section, the set of candidate representation learning functions Ψ is modeled as a set of kernel functions Ψ_{kernel}. Based on **Section** 3, a general VC dimension based kernel selection method is designed. For sake of discussion, we illustrate the method on binary classification problem.

4.1. Optimization Model

By substituting Ψ_{kernel} into formula (10), we obtain the following optimization problem (see **Appendix** A.3 for proof).

$$\begin{aligned} &\min_{\varphi \in \Psi_{\mathrm{kernel}}} \; f(\varphi) = 4\frac{g_1(\varphi)}{g_2(\varphi)} \\ &s.t. \begin{cases} g_1\left(\varphi\right) = \min\limits_{\mathbf{u}} \; \max\limits_{\mathbf{x}_i \in X} \\ \sigma\left(\mathbf{u}\right)^T \mathbf{K}^{\varphi} \sigma\left(\mathbf{u}\right) - 2\mathbf{K}^{\varphi}_{[i,:]}\sigma\left(\mathbf{u}\right) + k^{\varphi}_{ii} \\ g_2\left(\varphi\right) = \min\limits_{\mathbf{v},\mathbf{w}} \; \begin{pmatrix} \sigma\left(\mathbf{v}\right) \\ \sigma\left(\mathbf{w}\right) \end{pmatrix}^T \hat{\mathbf{K}}^{\varphi} \begin{pmatrix} \sigma\left(\mathbf{v}\right) \\ \sigma\left(\mathbf{w}\right) \end{pmatrix} \end{cases} \end{aligned} , \tag{11}$$

where $\mathbf{K}^{\varphi}$ and $\hat{\mathbf{K}}^{\varphi}$ are given in formula (29) in **Appendix** A.3, $\mathbf{K}^{\varphi}_{[i,:]}$ is the i-th row of matrix $\mathbf{K}^{\varphi}$, and $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ and σ are same as formula (10).

4.2. Model Solving

In formula (11), the feasible domain Ψ_{kernel} is a finite set. $\forall \varphi \in \Psi_{\mathrm{kernel}}$, the objective function $f(\varphi)$ can be calculated efficiently by the method given in **Section** 3.4. So it can be solved efficiently. See **Appendix** B.2 for details.

5. VC Dimension base DNN Boosting Framework

For a multiple classification problem, let $\mathcal{Y} = \{1, 2, \cdots, K\}$, $K > 2$ be output space. Let $\mathbf{X}$ be matrix composed of all training samples. $\forall p \in \mathcal{Y}$, let $\mathbf{X}_p$ be matrix composed of training samples that come from class p.

In this section, the representation learning function is mod-

eled as a DNN $\varphi : \mathcal{X} \to \mathcal{X}'$, $\mathcal{X}' \subseteq \mathbb{R}^{d'}$ with learnable parameters Θ. Based on **Section** 3, a general DNN boosting framework is proposed. An overall flowchart of the framework is shown in **Figure** 1.

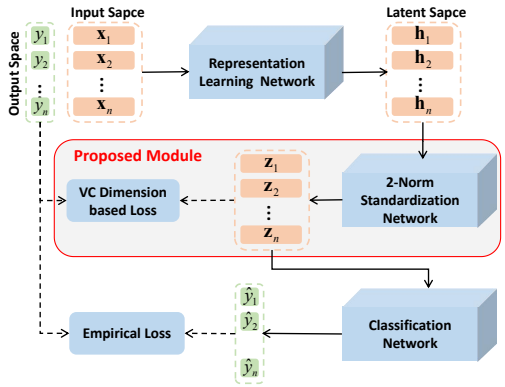

Figure 1. The flowchart of the VC dimension based DNN boosting framework. The proposed module is marked with a solid red box.

5.1. Optimization Model

$\forall \mathbf{x} \in \mathcal{X}$, let $\varphi(\mathbf{x};\Theta)$ be latent representation of sample $\mathbf{x}$. By substituting $\varphi(\cdot;\Theta)$ into formula (10), we have

$$\min_{\Theta} \ f(\Theta) = \sum_{p,q\in\mathcal{Y},p<q} 4\frac{g_1(\Theta)}{g_2^{(p,q)}(\Theta)}$$
$$s.t. \begin{cases} g_1(\Theta) = \min\limits_{\mathbf{u}} \ \max\limits_{\mathbf{x}_i\in X} \\ \|\varphi(\mathbf{x}_i;\Theta) - \varphi(\mathbf{X};\Theta)\sigma(\mathbf{u})\|_2^2 \\ \forall p,q\in\mathcal{Y}, p<q, \ g_2^{(p,q)}(\Theta) = \min\limits_{\mathbf{v},\mathbf{w}} \\ \|\varphi(\mathbf{X}_p;\Theta)\sigma(\mathbf{v}) - \varphi(\mathbf{X}_q;\Theta)\sigma(\mathbf{w})\|_2^2 \end{cases} . \quad (12)$$

To make formula (12) easier to solve, we introduce 2-norm standardization network, i.e., $\forall \mathbf{h} \in \mathbb{R}^{d'}$, $\lambda(\mathbf{h}) = \frac{\mathbf{h}}{\|\mathbf{h}\|_2}$. $\forall \mathbf{x} \in \mathcal{X}$, $\|\lambda(\varphi(\mathbf{x};\Theta))\|_2 = 1$, so the radius of latent space $\lambda(\varphi(\mathcal{X};\Theta))$ (see **Definition** A.7 in **Appendix** A.2) is not greater than 1. By substituting $\lambda(\cdot)$ into formula (12), we have

$$\min_{\Theta} \ -\sum_{p,q\in\mathcal{Y},p<q} g^{(p,q)}(\Theta)$$
$$s.t. \begin{cases} \forall p,q\in\mathcal{Y}, p<q, \ g^{(p,q)}(\Theta) = \min\limits_{\mathbf{v},\mathbf{w}} \\ \|\lambda(\varphi(\mathbf{X}_p;\Theta))\sigma(\mathbf{v}) - \lambda(\varphi(\mathbf{X}_q;\Theta))\sigma(\mathbf{w})\|_2^2 \end{cases} . \quad (13)$$

In order to achieve end-to-end learning, we add a linear classification network at the end of $\lambda(\varphi(\cdot;\Theta))$ and introduce empirical loss $\mathcal{L}$ into formula (13). The final optimization problem can be written as follows

$$\min_{\Theta,\mathbf{W},\mathbf{b}} \ \mathcal{L}(\Theta,\mathbf{W},\mathbf{b}) + \gamma_{\mathrm{VC}}\mathcal{L}_{\mathrm{VC}}(\Theta)$$
$$s.t. \begin{cases} \mathcal{L}(\Theta,\mathbf{W},\mathbf{b}) = \\ \frac{1}{n}\sum_{i=1}^{n} \ell_{\mathrm{CE}}(\sigma(\mathbf{W}\lambda(\varphi(\mathbf{x}_i;\Theta))+\mathbf{b}), y_i) \\ \mathcal{L}_{\mathrm{VC}}(\Theta) = -\frac{2}{|\mathcal{Y}|(|\mathcal{Y}|-1)}\sum_{p,q\in\mathcal{Y},p<q} g^{(p,q)}(\Theta) \\ \forall p,q\in\mathcal{Y}, p<q, \ g^{(p,q)}(\Theta) = \min\limits_{\mathbf{v},\mathbf{w}} \\ \|\lambda(\varphi(\mathbf{X}_p;\Theta))\sigma(\mathbf{v}) - \lambda(\varphi(\mathbf{X}_q;\Theta))\sigma(\mathbf{w})\|_2^2 \end{cases} , \quad (14)$$

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{Y}|\times d'}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{Y}|}$ are learnable parameters of linear classification network, ℓ_{CE} is cross entropy loss, $\mathcal{L}_{\mathrm{VC}}$ is VC dimension based loss, $\gamma_{\mathrm{VC}} > 0$ is trade-off parameter, and $\mathbf{v}$, $\mathbf{w}$ are vectors with adaptive length.

5.2. Model Solving

Given a batch of training samples, the optimization problem $g^{(p,q)}(\Theta)$ in formula (14) can be solved efficiently by the method given in **Section** 3.4. So the total loss $\mathcal{L}(\Theta,\mathbf{W},\mathbf{b}) + \gamma_{\mathrm{VC}}\mathcal{L}_{\mathrm{VC}}(\Theta)$ in formula (14) can be optimized efficiently by stochastic gradient descent with mini-batch even if the size of the training samples is large. See **Appendix** B.3 for details.

5.3. Related Work

In DeepLAD (Dorfer et al., 2016), a loss is designed based on the criteria of maximizing inter-class scatter and minimizing intra-class scatter to train DNNs. Although the loss is simple and intuitive, there is no theoretical analysis in the literature on whether it is conducive to reduce generalization error. In **Section** 6.3, we validate the advantage of the proposed method by comparison experiment.

6. Experiments

6.1. Verifying Theoretical Results

This section demonstrates the effectiveness of the representation learning framework proposed in **Section** 3.4.

6.1.1. Experimental Settings

Data set A 2-dimension data set Taichi is designed. The ground truth distribution of Taichi data set is shown in Figure 2a. The training data set consists of 628 positive samples and 629 negative samples, as shown in Figure 2b. The test data set consists of 251,312 positive samples and 251,313 negative samples.

Candidate functions Gaussian kernel $\varphi(\mathbf{x}_i;\delta)^T\varphi(\mathbf{x}_j;\delta) = \exp\left(\delta\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right)$, $\delta < 0$ is used. And there are 2000 kernel parameters, i.e., $\delta_k = \delta_{\min} + (k-1)\frac{\delta_{\max}-\delta_{\min}}{2000}$, $k = 1, 2, \cdots, 2000$, where $\delta_{\min} = -200$, $\delta_{\max} = -10^{-5}$.

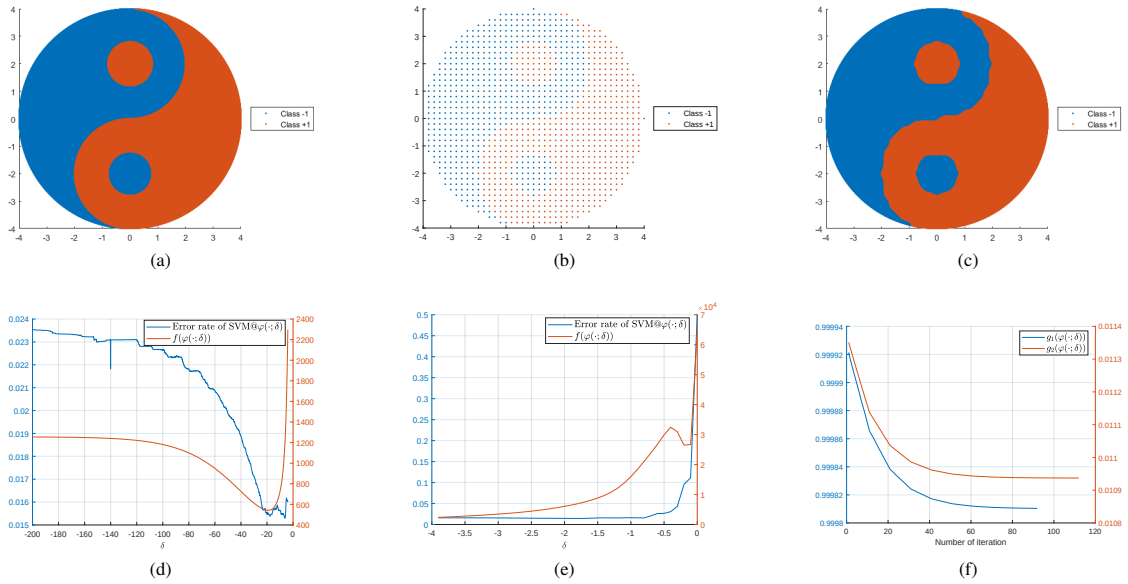

Figure 2. The experiments on Taichi data set. (a) The ground truth distribution. (b) The training data set. (c) The classification hyperplane of SVM@$\varphi(\cdot;\delta^*)$. (d) The error rate of SVM@$\varphi(\cdot;\delta)$ and the objective function value $f(\varphi(\cdot;\delta))$ in formula (10) with $\delta \in [-200, -4]$. (e) It's the same as (d), except for $\delta \in (-4, -10^{-5}]$. (f) The values of $g_1(\varphi(\cdot;\delta^*))$ and $g_2(\varphi(\cdot;\delta^*))$ in formula (10) after each iteration.

Basic learner The SVM (Cortes & Vapnik, 1995) with Gaussian kernel is selected as the basic learner. Let SVM@$\varphi(\cdot;\delta)$ denote the SVM classifier trained on the training data set with kernel function $\varphi(\cdot;\delta)$.

See **Appendix** C.1 for more experimental details.

6.1.2. Experimental Result and Analysis

We show the experimental results in **Figure** 2. We have

(a) **Figure** 2d and 2e show that the objective function value (red curve) $f(\varphi(\cdot;\delta))$ in formula (10) on training set and error rate (blue curve) on test set under different kernel parameters. It is observed that their change tendency is consistent, which demonstrates that the proposed method can be used for effectively estimating the generalization error of representation learning function $\varphi(\cdot;\delta)$.

(b) **Figure** 2c shows the classification hyperplane of SVM@$\varphi(\cdot;\delta^*)$, where δ^* is parameter corresponding to the lowest point of the red curve in **Figure** 2d. We can observe that the classification hyperplane with parameter δ^* is close to the ground truth distribution.

(c) **Figure** 2f shows the convergence of the objective functions $g_1(\cdot)$ (blue curve) and $g_2(\cdot)$ (red curve) in formula (10). The result shows the both objective functions can converge after a few iterations.

Therefore, the criterion in formula (10) can be used for effectively estimating the generalization error of representation learning function $\varphi(\cdot;\delta)$ and the corresponding optimization problems can be solved effectively and efficiently by the proposed **Algorithm** 1 and 2 in **Appendix** B.1.

6.2. Kernel Selection Experiment

This section demonstrates the effectiveness of the kernel selection method proposed in **Section** 4.

6.2.1. Experimental Settings

Data sets There are 15 binary classification data sets (see Table 5 in **Appendix** C.2 for details). For each data set, 80% samples are selected randomly as training data set and the remaining samples are selected as test data set. The ratio between positive samples and negative samples in training data set (and test data set) is equal to the whole data set.

Candidate kernel functions In this section, Gaussian kernel $\varphi(\mathbf{x}_i;\delta)^T \varphi(\mathbf{x}_j;\delta) = \exp\left(\delta \|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right)$, $\delta < 0$ is used. And there are 25 kernel parameters, i.e., $\delta_k = -2^{9-k}$, $k = 1, 2, \cdots, 25$, are used on each data set.

Basic learner The SVM (Cortes & Vapnik, 1995) with Gaussian kernel is selected as the basic learner. Let SVM@$\varphi(\cdot;\delta)$ denote the SVM classifier trained on the training data set with kernel function $\varphi(\cdot;\delta)$.

Comparison method Cross validation (Bengio & Grandvalet, 2004; Rodríguez et al., 2010; Jiang & Wang, 2017) is a commonly used method for kernel selection. 5-fold cross

validation is used to estimate the generalization performance of each candidate kernel function. To reduce randomness, 5 different 5-fold cross validations (denoted as T1, T2, $\cdots$, T5) are conducted. Let δ_{CV}^* be the selected parameter.

Proposed method For each data set, the training data set and the set of candidate kernel functions are fed to the **Algorithm** 3 in **Appendix** B. Let δ_{opt} be the selected parameter.

Oracle and evaluation criteria Given a learning task (a data set), the goal of learning is to minimize generalization error, i.e., error rate on test set. In this experiment, the error rate of SVM@$\varphi(\cdot;\delta_k)$, $k = 1, 2, \cdots, 25$ on test set are used as oracle to estimate the generalization performance of each candidate parameter. So the oracle-rank of every candidate parameter $rank(\delta_k) \in \{1, 2, \cdots, 25\}$ can be calculated. Then, $rank(\delta_{\mathrm{CV}}^*)$ and $rank(\delta_{\mathrm{opt}})$ are used to compare the performance of the corresponding methods.

See **Appendix** C.2 for more experimental details.

Table 2. The oracle-ranks of different methods.

ID	5-Fold Cross Validation						Our
	T1	T2	T5	T4	T5	mean	
D1	2	2	2	2	2	2.00	**1**
D2	**1**	**1**	**1**	2	2	1.40	**1**
D3	3	3	2	3	3	2.80	**1**
D4	3	2	3	3	2	2.60	**1**
D5	2	2	2	2	2	2.00	**1**
D6	**1**	3	3	3	**1**	2.20	**1**
D7	3	3	3	3	3	3.00	**2**
D8	5	5	5	6	4	5.00	**2**
D9	**1**	**1**	3	**1**	**1**	1.40	**1**
D10	**1**	4	4	**1**	**1**	2.20	**1**
D11	3	4	2	4	2	3.00	**1**
D12	3	3	**1**	**1**	3	2.20	**1**
D13	**1**	2	**1**	3	**1**	1.60	**1**
D14	**1**	2	2	2	2	1.80	**1**
D15	**1**	2	2	2	2	1.80	**1**

6.2.2. Experimental Result and Analysis

We conduct two groups of experiments from the perspective of generalization performance and time efficiency.

First, the oracle-rank of different methods is used for evaluating the generalization performance. The experimental results on 15 data sets are presented in **Table** 2. It can be observed that the oracle-rank of the proposed kernel selection method is superior to the means of 5-fold cross validation on the all data sets. Therefore, the generalization performance of the proposed method is superior to 5-fold cross validation. In addition, cross validation is inherently of randomness, while the proposed method is of no randomness.

Second, we report the running times of 5-fold cross validation and the proposed method for evaluating a kernel function in **Table** 3. The results show that with the increase of the number of training samples, the time efficiency of the proposed method is significantly higher than of the 5-fold cross validation. See **Appendix** B.2.1 for detailed analysis.

Therefore, the proposed method can effectively and efficiently select kernel with good generalization performance for kernel based method.

Table 3. The running times of different methods (ms).

ID	Training Samples	5-Fold Cross Validation	**Our**	Ratio
D1	8	**1.78**	5.96	0.30
D2	64	**2.97**	4.37	0.68
D3	86	**3.95**	4.54	0.87
D4	136	5.94	**4.39**	**1.35**
D5	146	7.41	**6.14**	**1.21**
D6	157	8.67	**5.71**	**1.52**
D7	236	15.43	**6.28**	**2.46**
D8	245	10.92	**5.97**	**1.83**
D9	456	58.78	**5.90**	**9.96**
D10	467	41.87	**5.16**	**8.12**
D11	486	71.53	**5.82**	**12.29**
D12	553	74.16	**18.33**	**4.05**
D13	560	32.22	**5.68**	**5.67**
D14	5,921	8,406.90	**52.56**	**159.96**
D15	15,217	33,147.24	**22.08**	**1,501.23**

6.3. DNN Boosting Experiment

This section demonstrates the effectiveness of the DNN boosting framework proposed in **Section** 5.

6.3.1. Experimental Settings

Data sets The MNIST and CIFAR10 are used in this section. In this section, 10, 20, $\cdots$, 60 samples are randomly selected from each class in the training set to train the models and the 10,000 test samples are used to evaluate the models.

Basic DNNs There are 5 DNNs used in this section, i.e., FCNet3, LeNet (LeCun et al., 1989), ResNet18 (He et al., 2016), ResNet50 (He et al., 2016) and ViT-Base (Dosovitskiy et al., 2021). Among them, FCNet3 hardly uses domain knowledge, LeNet uses less domain knowledge, ResNet18, ResNet50, and ViT-Base use more domain knowledge.

Comparison framework The DeepLAD (Dorfer et al., 2016) is selected as the comparison framework.

$\forall Net \in \{$ FCNet3, LeNet, ResNet18, ResNet50, ViT-Base$\}$, let Net, Net+LDA, and Net+Our be the basic DNN, the DNN embedded into DeepLDA, and the DNN embedded into the proposed framework, respectively.

See **Appendix** C.3 for more experimental details.

Table 4. The accuracy of different methods on test data set (%).

DATA SET	METHOD	THE NUMBER OF TRAINING SAMPLES IN EACH CLASS					
		10	20	30	40	50	60
MNIST	FCNET3	75.91	83.49	85.76	87.21	87.93	88.87
	FCNET3+LDA	35.26	41.70	45.07	43.91	42.65	44.57
	FCNET3+OUR	**78.74**	**85.14**	**87.27**	**88.44**	**89.60**	**90.25**
	LENET	67.34	71.66	82.14	83.67	84.13	84.53
	LENET+LDA	34.96	34.73	35.46	33.32	35.19	36.61
	LENET+OUR	**71.36**	**74.73**	**83.99**	**94.04**	**94.23**	**94.78**
	RESNET18	76.70	84.57	89.42	90.55	91.46	91.74
	RESNET18+LDA	41.51	43.13	46.68	44.00	47.10	49.19
	RESNET18+OUR	**85.35**	**90.22**	**92.81**	**93.38**	**94.16**	**94.28**
	RESNET50	72.43	81.53	87.43	89.36	88.94	89.41
	RESNET50+LDA	22.51	38.12	38.55	39.60	39.87	40.66
	RESNET50+OUR	**78.09**	**85.74**	**88.13**	**89.83**	**90.84**	**91.15**
	VIT-BASE	71.16	78.00	82.91	84.22	85.69	87.23
	VIT-BASE+LDA	53.98	56.83	58.07	58.83	59.01	59.88
	VIT-BASE+OUR	**72.47**	**80.50**	**86.14**	**87.68**	**88.96**	**89.56**
CIFAR10	FCNET3	21.53	24.81	26.41	29.01	29.72	29.54
	FCNET3+LDA	17.63	19.54	21.43	21.06	21.06	20.55
	FCNET3+OUR	**24.54**	**27.48**	**30.11**	**31.34**	**32.62**	**32.25**
	LENET	13.39	16.99	18.51	18.66	20.31	22.31
	LENET+LDA	14.78	14.20	15.40	15.73	15.30	15.39
	LENET+OUR	**22.51**	**23.13**	**26.89**	**29.19**	**29.60**	**30.47**
	RESNET18	24.63	27.87	32.53	33.83	34.21	35.06
	RESNET18+LDA	16.26	15.40	16.63	18.26	18.81	19.29
	RESNET18+OUR	**26.60**	**30.77**	**33.98**	**35.78**	**36.64**	**36.87**
	RESNET50	22.99	26.32	29.74	30.77	30.45	30.51
	RESNET50+LDA	15.27	18.01	17.92	15.64	18.14	20.60
	RESNET50+OUR	**23.68**	**27.68**	**30.64**	**32.03**	**32.21**	**32.31**
	VIT-BASE	21.28	22.65	23.96	25.14	25.20	25.95
	VIT-BASE+LDA	19.30	19.09	20.41	20.30	21.05	20.33
	VIT-BASE+OUR	**22.81**	**24.17**	**26.52**	**29.10**	**27.34**	**27.98**

6.3.2. EXPERIMENTAL RESULT AND ANALYSIS

We report the test accuracy of all methods with different number of the training samples in **Table** 4. We have

(a) In all cases, $\forall Net \in \{$ FCNet3, LeNet, ResNet18, ResNet50, ViT-Base$\}$, the performances of Net+Our is superior to Net. These results demonstrate the universality and effectiveness of the proposed framework.

(b) When the number of the training samples is small, the proposed framework can boost the performance of basic DNNs. Therefore, the proposed framework can alleviate the dependence of DNN on a large number labeled samples to a certain extent.

(c) On MNIST data set, we have (c1) The performance of ResNet18 is superior to FCNet3 and LeNet in all cases because it is a well-designed network architecture that uses more domain knowledge. (c2) The performance of FCNet3+Our is superior to LeNet with 10, 20, ..., 60 training samples. (c3) The performance of LeNet+Our is superior to ResNet18 with 40, 50, and 60 training samples. Similar results can also be seen on the CIFAR10 data set, e.g, FCNet3+Our is superior to LeNet 10, 20, ..., 60 training samples. These results show that the proposed framework can achieve better performance with less or no domain knowledge.

(d) On both two data sets, ResNet18 gets the highest performance among 5 basic DNNs. The network architecture of ResNet18 is more complex than FCNet3 and LeNet, but simpler than ResNet50 and ViT-Base. These results show that the complexity of the network architecture needs to match the task in order to achieve good performance. The proposed framework can improve their performance regardless of whether the network architecture matches the task.

(e) In all cases, $\forall Net \in \{$ FCNet3, LeNet, ResNet18, ResNet50, ViT-Base$\}$ the performances of Net+Our is superior to Net+LDA. These results show that the proposed method is superior to DeepLDA as a boosting framework. This is because DeepLDA needs to estimate the covariance

matrix of the latent representations. When the number of the training samples is small, it is difficult to obtain accurate results. On the contrary, the radius and convex hull of the samples can better depict the characteristics of the latent representations, thus estimating these variables can better approximate the generalization error.

In summary, the proposed boosting framework is universal and effective, and can alleviate some problems of DNN.

7. Conclusion

In this paper, a VC dimension based criterion is designed. The criterion has clear geometric meaning and can estimate generalization error of representation learning function effectively and efficiently. Based on the criterion, we propose a general representation learning framework which aims to reduce generalization error of representation learning. And the framework has been successfully applied to kernel selection and DNN boosting. As a general framework, it is promising to combine with other machine learning methods and improve their generalization performance.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (under grant 2020AAA0106100) and the National Natural Science Foundation of China (Nos. 62006147, U21A20473).

References

Bengio, Y. and Grandvalet, Y. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 1 edition, 2004.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Advances in Neural Information Processing Systems, Virtual Event*, pp. 1877–1901, 2020.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European Conference on Computer Vision, Glasgow, UK, Proceedings, Part I*, pp. 213–229, 2020.

Chiu, C., Qin, J., Zhang, Y., Yu, J., and Wu, Y. Self-supervised learning with random-projection quantizer for speech recognition. In *International Conference on Machine Learning, Baltimore, Maryland, USA*, pp. 3915–3924, 2022.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Cuadros, X. S., Zappella, L., and Apostoloff, N. Self-conditioning pre-trained language models. In *International Conference on Machine Learning, Baltimore, Maryland, USA*, pp. 4455–4473, 2022.

Cui, J. and Liang, J. Fuzzy learning machine. In *Advances in Neural Information Processing Systems, Virtual Event*, pp. 36693–36705, 2022.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA*, pp. 4171–4186, 2019.

Dorfer, M., Kelz, R., and Widmer, G. Deep linear discriminant analysis. In *International Conference on Learning Representations, San Juan, Puerto Rico*, 2016.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations, Virtual Event*, 2021.

Goodfellow, I. J., Bengio, Y., and Courville, A. C. *Deep Learning*. MIT Press, 1 edition, 2016.

Gulati, A., Qin, J., Chiu, C., Parmar, N., Zhang, Y., Yu, J., Han, W., Wang, S., Zhang, Z., Wu, Y., and Pang, R. Conformer: Convolution-augmented transformer for speech recognition. In *Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China*, pp. 5036–5040, 2020.

Hanson, S. J. and Pratt, L. Y. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems, Denver, Colorado, USA*, pp. 177–185, 1988.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA*, pp. 770–778, 2016.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. B. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA*, pp. 15979–15988, 2022.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Jiang, G. and Wang, W. Error estimation based on variance analysis of k-fold cross-validation. *Pattern Recognition*, 69:94–106, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, San Diego, CA, USA*, 2015.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Annual Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, USA*, pp. 1106–1114, 2012.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems, Denver, Colorado, USA*, pp. 396–404, 1989.

LeCun, Y., Bengio, Y., and Hinton, G. E. Deep learning. *Nature*, 521(7553):436–444, 2015.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations, New Orleans, LA, USA*, 2019.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10): 1872–1897, 2020.

Rodríguez, J. D., Martínez, A. P., and Lozano, J. A. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):569–575, 2010.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. Deep convolutional neural networks for LVCSR. In *IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada*, pp. 8614–8618, 2013.

Schölkopf, B., Smola, A. J., and Müller, K. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations, San Diego, CA, USA*, 2015.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Annual Conference on Neural Information Processing Systems, Montreal, Quebec, Canada*, pp. 3104–3112, 2014.

Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, 2 edition, 1999.

Zhang, R. and Rudnicky, A. I. A large scale clustering scheme for kernel k-means. In *International Conference on Pattern Recognition, Quebec, Canada*, pp. 289–292, 2002.

A. Proofs

A.1. Proof of Theorem 3.1

Step 1: Some useful definitions and lemmas are given as follows.

Definition A.1 (Page 34 of (Boyd & Vandenberghe, 2004))**.** Given a set $X \subset \mathbb{R}^d$, the convex hull of X is defined as

$$ch\left(X\right) = \left\{ \sum_{i=1}^{|X|} \alpha_i \mathbf{x}_i \,\middle|\, \boldsymbol{\alpha} \in \triangle^{|X|} \right\}.$$

Obviously, $X \subseteq ch\left(X\right)$, and $ch\left(X\right)$ is a convex set.

Definition A.2. Given two sets $X_+ = \left\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\right\}_{i=1}^{n_+}$ and $X_- = \left\{\mathbf{x}_j | \mathbf{x}_j \in \mathbb{R}^d\right\}_{j=n_++1}^{n_++n_-}$, $n = n_+ + n_-$. The distance in $\|\cdot\|_2$ between the convex hulls $ch\left(X_+\right)$ and $ch\left(X_-\right)$, denoted as $dis\left(ch\left(X_+\right), ch\left(X_-\right)\right)$, is defined as follows

$$\begin{aligned} dis^2\left(ch\left(X_+\right), ch\left(X_-\right)\right) = \quad & \min_{\mathbf{a},\,\mathbf{b}} \|\mathbf{a}-\mathbf{b}\|_2^2 \\ & s.t.\ \ \mathbf{a} \in ch\left(X_+\right) \\ & \quad\ \ \mathbf{b} \in ch\left(X_-\right) \end{aligned} \cdot \tag{15}$$

Lemma A.3. *The optimization problem* (15) *is a convex optimization problem.*

Proof.

$$\begin{aligned} dis^2\left(ch\left(X_+\right), ch\left(X_-\right)\right) &= \min_{\mathbf{a}\in ch(X_+),\ \mathbf{b}\in ch(X_-)} \|\mathbf{a}-\mathbf{b}\|_2^2 \\ &= \min_{\boldsymbol{\alpha}\in\triangle^{n_+},\ \boldsymbol{\beta}\in\triangle^{n_-}} \|\mathbf{X}_+\boldsymbol{\alpha} - \mathbf{X}_-\boldsymbol{\beta}\|_2^2 \quad \text{// see } \textbf{Definition } A.1 \\ &= \min_{\boldsymbol{\alpha}\in\triangle^{n_+},\ \boldsymbol{\beta}\in\triangle^{n_-}} \left(\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T\right) \mathbf{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix} \end{aligned}, \tag{16}$$

where $\mathbf{K} = \left(\mathbf{X}_+, -\mathbf{X}_-\right)^T \left(\mathbf{X}_+, -\mathbf{X}_-\right) \in \mathbb{R}^{n\times n}$.

Obviously, $\mathbf{K}$ is a symmetric positive semidefinite matrix. So (a1) $\left(\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T\right) \mathbf{K} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}$ is convex function w.r.t. $\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \end{pmatrix}$.

At the same time, (a2) both $\triangle^{n_+}$ and $\triangle^{n_-}$ involve only linear constraints.

Combining (a1) and (a2), the above formula is a convex optimization problem. □

Step 2: The margin of hyperplane described in **Definition** 2.2 is bounded by the distance between two convex hulls described in **Definition** A.2, see the following lemma.

Lemma A.4. *Given two sets* $X_+ = \left\{\boldsymbol{x}_i | \boldsymbol{x}_i \in \mathbb{R}^d\right\}_{i=1}^{n_+}$ *and* $X_- = \left\{\boldsymbol{x}_j | \boldsymbol{x}_j \in \mathbb{R}^d\right\}_{j=n_++1}^{n_++n_-}$, $n = n_+ + n_-$. *Let* $dis\left(ch\left(X_+\right), ch\left(X_-\right)\right)$ *be the distance between* $ch(X_+)$ *and* $ch(X_-)$ *given in* ***Definition*** *A.2. If* $dis\left(ch\left(X_+\right), ch\left(X_-\right)\right) > 0$, *then* $\exists \boldsymbol{\omega} \in \mathbb{R}^d$, $\|\boldsymbol{\omega}\|_2 = 1$, $\exists b \in \mathbb{R}$, *such that*
(i) $\forall \hat{\boldsymbol{p}} \in ch(X_+)$, $\boldsymbol{\omega}^T\hat{\boldsymbol{p}} + b \geq M$,
(ii) $\forall \hat{\boldsymbol{n}} \in ch(X_-)$, $\boldsymbol{\omega}^T\hat{\boldsymbol{n}} + b \leq -M$,
where $M = \frac{dis(ch(X_+), ch(X_-))}{2}$.

Proof. Let

$$(\mathbf{p}, \mathbf{n}) = \underset{\mathbf{a}\in ch(X_+), \mathbf{b}\in ch(X_-)}{arg\,min} \|\mathbf{a}-\mathbf{b}\|_2^2, \tag{17a}$$

$$\boldsymbol{\omega} = \frac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}, \tag{17b}$$

$$b = -\boldsymbol{\omega}^T \frac{\mathbf{p}+\mathbf{n}}{2}. \tag{17c}$$

Obviously, $\|\boldsymbol{\omega}\|_2 = 1$. According to **Definition** A.2, we have $dis\left(ch\left(X_+\right), ch\left(X_-\right)\right) = \|\mathbf{p}-\mathbf{n}\|_2$. So $M = \frac{1}{2}dis\left(ch\left(X_+\right), ch\left(X_-\right)\right) = \frac{1}{2}\|\mathbf{p}-\mathbf{n}\|_2$

(A) Proof of (i).

$\forall \hat{\mathbf{p}} \in ch(X_+)$, if $\hat{\mathbf{p}} = \mathbf{p}$, then

$$\begin{aligned} \boldsymbol{\omega}^T\hat{\mathbf{p}} + b &= \boldsymbol{\omega}^T\mathbf{p} + b \\ &= \boldsymbol{\omega}^T\mathbf{p} - \boldsymbol{\omega}^T\frac{\mathbf{p}+\mathbf{n}}{2} \\ &= \boldsymbol{\omega}^T\left(\mathbf{p} - \frac{\mathbf{p}+\mathbf{n}}{2}\right) \\ &= \boldsymbol{\omega}^T\left(\frac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= \left(\frac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T\left(\frac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= \frac{1}{2}\|\mathbf{p}-\mathbf{n}\|_2 = M \end{aligned} . \tag{18}$$

$\forall \hat{\mathbf{p}} \in ch(X_+)$, if $\hat{\mathbf{p}} \neq \mathbf{p}$, then

$$\begin{aligned} \boldsymbol{\omega}^T\hat{\mathbf{p}} + b &= \boldsymbol{\omega}^T\hat{\mathbf{p}} - \boldsymbol{\omega}^T\frac{\mathbf{p}+\mathbf{n}}{2} \\ &= \boldsymbol{\omega}^T\left(\hat{\mathbf{p}} - \frac{\mathbf{p}+\mathbf{n}}{2}\right) \\ &= \boldsymbol{\omega}^T\left(\left(\hat{\mathbf{p}} - \frac{\mathbf{p}+\mathbf{n}}{2} - \frac{\mathbf{p}-\mathbf{n}}{2}\right) + \frac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= \boldsymbol{\omega}^T\left(\hat{\mathbf{p}} - \mathbf{p} + \frac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= \left(\frac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T\left(\hat{\mathbf{p}} - \mathbf{p} + \frac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= \left(\frac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T\left(\hat{\mathbf{p}} - \mathbf{p}\right) + \frac{\|\mathbf{p}-\mathbf{n}\|_2}{2} \\ &= \left(\frac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T\left(\hat{\mathbf{p}} - \mathbf{p}\right) + M \end{aligned} . \tag{19}$$

According to formula (19), in order to proof that $\boldsymbol{\omega}^T\hat{\mathbf{p}} + b \geq M$, we need to proof that $\left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right) \geq 0$. Next, we will proof $\left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right) \geq 0$ by reduction to absurdity.

Assume that $\left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right) < 0$.

Let $\mathbf{q} = (1-\theta)\mathbf{p} + \theta\hat{\mathbf{p}}$, where $\theta \in (0,1)$. $\because \hat{\mathbf{p}} \in ch(X_+)$, $\mathbf{p} \in ch(X_+)$, and $ch(X_+)$ is a convex set, $\therefore \mathbf{q} \in ch(X_+)$. And then, we have

$$\begin{aligned} \|\mathbf{n}-\mathbf{q}\|_2^2 &= \|\mathbf{n} - (1-\theta)\mathbf{p} - \theta\hat{\mathbf{p}}\|_2^2 \\ &= \|(\mathbf{n}-\mathbf{p}) + \theta\left(\mathbf{p}-\hat{\mathbf{p}}\right)\|_2^2 \\ &= \|\mathbf{n}-\mathbf{p}\|_2^2 + \theta^2\|\mathbf{p}-\hat{\mathbf{p}}\|_2^2 + 2\theta\left(\mathbf{n}-\mathbf{p}\right)^T\left(\mathbf{p}-\hat{\mathbf{p}}\right) \\ &= \|\mathbf{p}-\mathbf{n}\|_2^2 + \theta^2\|\hat{\mathbf{p}}-\mathbf{p}\|_2^2 + 2\theta\left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right) \end{aligned} . \tag{20}$$

Given formula (20), let $g(\theta) = \theta^2\|\hat{\mathbf{p}}-\mathbf{p}\|_2^2 + 2\theta\left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right)$ be the quadratic function w.r.t θ. And then we have
(a1) $\because \hat{\mathbf{p}} \neq \mathbf{p}, \therefore \|\hat{\mathbf{p}}-\mathbf{p}\|_2^2 > 0$,
(a2) $g(0) = 0$,
(a3) $g\left(-\frac{2(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{p}}-\mathbf{p})}{\|\hat{\mathbf{p}}-\mathbf{p}\|_2^2}\right) = 0$,
(a4) $\because \left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right) < 0, \therefore -\frac{2(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{p}}-\mathbf{p})}{\|\hat{\mathbf{p}}-\mathbf{p}\|_2^2} > 0$.

Combining (a1)-(a4), we have $\forall\theta' \in (0,1)\bigcap\left(0, -\frac{2(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{p}}-\mathbf{p})}{\|\hat{\mathbf{p}}-\mathbf{p}\|_2^2}\right)$, $g(\theta') < 0$. And let $\mathbf{q}' = (1-\theta')\mathbf{p} + \theta'\hat{\mathbf{p}} \in ch(X_+)$, then $\|\mathbf{n}-\mathbf{q}'\|_2^2 = \|\mathbf{p}-\mathbf{n}\|_2^2 + g(\theta') < \|\mathbf{p}-\mathbf{n}\|_2^2$. That is to say, $\exists\mathbf{q}' \in ch(X_+)$, such that $\|\mathbf{n}-\mathbf{q}'\|_2^2 < \|\mathbf{p}-\mathbf{n}\|_2^2$, which conflicts with formula (17a). So the assumption is false, and we have (a5) $\left(\mathbf{p}-\mathbf{n}\right)^T\left(\hat{\mathbf{p}}-\mathbf{p}\right) \geq 0$.

According to (a5) and formula (19), we have (a6) $\boldsymbol{\omega}^T\hat{\mathbf{p}} + b \geq M$.

Combining formula (18) and (a6), (i) is proofed.

(B) Proof of (ii).

$\forall \hat{\mathbf{n}} \in ch(X_+)$, if $\hat{\mathbf{n}} = \mathbf{n}$, then

$$\begin{aligned} \boldsymbol{\omega}^T\hat{\mathbf{n}} + b &= \boldsymbol{\omega}^T\mathbf{n} + b \\ &= \boldsymbol{\omega}^T\mathbf{n} - \boldsymbol{\omega}^T\tfrac{\mathbf{p}+\mathbf{n}}{2} \\ &= \boldsymbol{\omega}^T\left(\mathbf{n} - \tfrac{\mathbf{p}+\mathbf{n}}{2}\right) \\ &= -\boldsymbol{\omega}^T\left(\tfrac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= -\left(\tfrac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T\left(\tfrac{\mathbf{p}-\mathbf{n}}{2}\right) \\ &= -\tfrac{1}{2}\|\mathbf{p}-\mathbf{n}\|_2 = -M \end{aligned} . \tag{21}$$

$\forall \hat{\mathbf{n}} \in ch(X_+)$, if $\hat{\mathbf{n}} \neq \mathbf{n}$, then

$$\begin{aligned} \boldsymbol{\omega}^T\hat{\mathbf{n}} + b &= \boldsymbol{\omega}^T\hat{\mathbf{n}} - \boldsymbol{\omega}^T\tfrac{\mathbf{p}+\mathbf{n}}{2} \\ &= \boldsymbol{\omega}^T\left(\hat{\mathbf{n}} - \tfrac{\mathbf{p}+\mathbf{n}}{2}\right) \\ &= \boldsymbol{\omega}^T\left(\left(\hat{\mathbf{n}} - \tfrac{\mathbf{p}+\mathbf{n}}{2} - \tfrac{\mathbf{n}-\mathbf{p}}{2}\right) + \tfrac{\mathbf{n}-\mathbf{p}}{2}\right) \\ &= \boldsymbol{\omega}^T\left(\hat{\mathbf{n}} - \mathbf{n} + \tfrac{\mathbf{n}-\mathbf{p}}{2}\right) \\ &= \left(\tfrac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T\left(\hat{\mathbf{n}} - \mathbf{n} + \tfrac{\mathbf{n}-\mathbf{p}}{2}\right) \\ &= \left(\tfrac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T(\hat{\mathbf{n}} - \mathbf{n}) - \tfrac{\|\mathbf{p}-\mathbf{n}\|_2}{2} \\ &= \left(\tfrac{\mathbf{p}-\mathbf{n}}{\|\mathbf{p}-\mathbf{n}\|_2}\right)^T(\hat{\mathbf{n}} - \mathbf{n}) - M \end{aligned} . \tag{22}$$

According to formula (22), in order to proof that $\boldsymbol{\omega}^T\hat{\mathbf{n}} + b \leq -M$, we need to proof that $(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{n}}-\mathbf{n}) \leq 0$. Next, we will proof $(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{n}}-\mathbf{n}) \leq 0$ by reduction to absurdity.

Assume that $(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{n}}-\mathbf{n}) > 0$.

Let $\mathbf{o} = (1-\theta)\mathbf{n} + \theta\hat{\mathbf{n}}$, where $\theta \in (0,1)$. $\because \hat{\mathbf{n}} \in ch(X_-)$, $\mathbf{n} \in ch(X_-)$, and $ch(X_-)$ is a convex set, $\therefore \mathbf{o} \in ch(X_-)$. And then, we have

$$\begin{aligned} \|\mathbf{p}-\mathbf{o}\|_2^2 &= \|\mathbf{p} - (1-\theta)\mathbf{n} - \theta\hat{\mathbf{n}}\|_2^2 \\ &= \|(\mathbf{p}-\mathbf{n}) + \theta(\mathbf{n}-\hat{\mathbf{n}})\|_2^2 \\ &= \|\mathbf{p}-\mathbf{n}\|_2^2 + \theta^2\|\mathbf{n}-\hat{\mathbf{n}}\|_2^2 + 2\theta(\mathbf{p}-\mathbf{n})^T(\mathbf{n}-\hat{\mathbf{n}}) \end{aligned} . \tag{23}$$

Given formula (23), let $g(\theta) = \theta^2\|\mathbf{n}-\hat{\mathbf{n}}\|_2^2 + 2\theta(\mathbf{p}-\mathbf{n})^T(\mathbf{n}-\hat{\mathbf{n}})$ be the quadratic function w.r.t θ. And then we have
(b1) $\because \hat{\mathbf{n}} \neq \mathbf{n}, \therefore \|\mathbf{n}-\hat{\mathbf{n}}\|_2^2 > 0$,
(b2) $g(0) = 0$,
(b3) $g\left(-\frac{2(\mathbf{p}-\mathbf{n})^T(\mathbf{n}-\hat{\mathbf{n}})}{\|\mathbf{n}-\hat{\mathbf{n}}\|_2^2}\right) = 0$,
(b4) $\because (\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{n}}-\mathbf{n}) < 0, \therefore -\frac{2(\mathbf{p}-\mathbf{n})^T(\mathbf{n}-\hat{\mathbf{n}})}{\|\mathbf{n}-\hat{\mathbf{n}}\|_2^2} > 0$.

Combining (b1)-(b4), we have $\forall\theta' \in (0,1)\bigcap\left(0, -\frac{2(\mathbf{p}-\mathbf{n})^T(\mathbf{n}-\hat{\mathbf{n}})}{\|\mathbf{n}-\hat{\mathbf{n}}\|_2^2}\right)$, $g(\theta') < 0$. And let $\mathbf{o}' = (1-\theta')\mathbf{n} + \theta'\hat{\mathbf{n}} \in ch(X_-)$, then $\|\mathbf{p}-\mathbf{o}'\|_2^2 = \|\mathbf{p}-\mathbf{n}\|_2^2 + g(\theta') < \|\mathbf{p}-\mathbf{n}\|_2^2$. That is to say, $\exists\mathbf{o}' \in ch(X_-)$, such that $\|\mathbf{p}-\mathbf{o}'\|_2^2 < \|\mathbf{p}-\mathbf{n}\|_2^2$, which conflicts with formula (17a). So the assumption is false, and we have (b5) $(\mathbf{p}-\mathbf{n})^T(\hat{\mathbf{n}}-\mathbf{n}) \leq 0$.

According to (b5) and formula (22), we have (b6) $\boldsymbol{\omega}^T\hat{\mathbf{n}} + b \leq -M$.

Combining formula (21) and (b6), (ii) is proofed.

Combining (A) and (B), the lemma is proofed. □

Remark A.5. According to **Lemma** A.4, if $dis\left(ch\left(\varphi\left(X_+\right)\right), ch\left(\varphi\left(X_+\right)\right)\right) > 0$, then $\exists h \in \mathcal{H}\left(\varphi\left(X_+\right), \varphi\left(X_-\right)\right)$, such that $\forall\mathbf{x}_i \in X_+$, $h\left(\varphi\left(\mathbf{x}_i\right)\right) > 0$, and $\forall\mathbf{x}_j \in X_-$, $h\left(\varphi\left(\mathbf{x}_j\right)\right) < 0$, i.e., $\frac{n_{err}}{n} = 0$. That is to say, constraint $\frac{n_{err}}{n} \leq \varepsilon$ in formula (7) can be satisfied naturally.

Step 3: Based on the above conclusions, the proof of **Theorem** 3.1 is given as follows.

Proof. (a1)

$$\begin{aligned} M^2(X_+, X_-) &= \sup_{h \in \mathcal{H}(X_+, X_-)} g_{\mathrm{m}}^2(h) \\ &\geq \sup_{h = ph(\boldsymbol{\omega}, b)} g_{\mathrm{m}}^2(h) && // \ \boldsymbol{\omega}, b \ are \ defined \ in \ formula \ (17) \\ &\geq \tfrac{1}{4} dis^2(ch(X_+), ch(X_+)) && // \ according \ to \ \textbf{Lemma} \ A.4 \\ &= \tfrac{1}{4} \min_{\boldsymbol{\alpha} \in \triangle^{n_+}, \, \boldsymbol{\beta} \in \triangle^{n_-}} \|\mathbf{X}_+ \boldsymbol{\alpha} - \mathbf{X}_- \boldsymbol{\beta}\|_2^2. && // \ according \ to \ \textbf{Lemma} \ A.3 \end{aligned}$$

(a2) According to **Lemma** A.3, the $\min_{\boldsymbol{\alpha} \in \triangle^{n_+}, \, \boldsymbol{\beta} \in \triangle^{n_-}} \|\mathbf{X}_+ \boldsymbol{\alpha} - \mathbf{X}_- \boldsymbol{\beta}\|_2^2$ is a convex optimization problem.

Combining (a1) and (a2), the theorem is proofed. □

A.2. Proof of Theorem 3.2

Step 1: A useful definition is given as follows.

Definition A.6 (Page 30 of (Boyd & Vandenberghe, 2004))**.** Given $\mathbf{c} \in \mathbb{R}^d$ and $r > 0$, the ball in norm $\|\cdot\|_2$ is defined as $\mathrm{ball}(\mathbf{c}, r) = \{\mathbf{v} | \mathbf{v} \in \mathbb{R}^d, \|\mathbf{v} - \mathbf{c}\|_2 \leq r\}$, where $\mathbf{c}$ is called the center of the ball and r is called the radius of the ball. Obviously, ball($\mathbf{c}$, r) is a convex set.

Step 2: Based on **Definition** A.6, the radius $R(X)$ can be defined as follows.

Definition A.7. Given a set $X \subset \mathbb{R}^d$, the center and radius of X, denoted as $C(X)$ and $R(X)$, are defined as the center and radius of the minimum ball that can include X, respectively, i.e.,

$$(C(X), R(X)) = \arg\inf_{\mathbf{c} \in \mathbb{R}^d, \, r \in \mathbb{R}} \ r \ s.t. \ X \subseteq ball(\mathbf{c}, r). \tag{24}$$

The formula (24) is a non-trivial optimization problem, because constraints $\mathbf{c} \in \mathbb{R}^d$ and $X \subseteq ball(\mathbf{c}, r)$ are intractable for numerical optimization.

Step 3: To solve this problem, the properties of the optimal solution of formula (24) need to be revealed, see the following lemma.

Lemma A.8. *Given a set $X \subset \mathbb{R}^d$, the center of X belongs to the convex hull of X, i.e., if*

$$(\boldsymbol{c}^*, r^*) = \arg\inf_{\boldsymbol{c}_i \in \mathbb{R}^d, \, r \in \mathbb{R}} \ r \ s.t. \ X \subseteq ball(\boldsymbol{c}, r), \tag{25}$$

then $\boldsymbol{c}^* \in ch(X)$.

Proof. Reduction to absurdity. Assume that $\mathbf{c}^* \notin ch(X)$.

Let

$$\mathbf{x}^* = \arg\inf_{\mathbf{x} \in X} \|\mathbf{x} - \mathbf{c}^*\|_2, \tag{26a}$$

$$\mathbf{c} = \frac{\mathbf{x}^* + \mathbf{c}^*}{2}, \tag{26b}$$

$$\boldsymbol{\omega} = \frac{\mathbf{x}^* - \mathbf{c}^*}{\|\mathbf{x}^* - \mathbf{c}^*\|_2}, \tag{26c}$$

$$b_1 = -\boldsymbol{\omega}^T \mathbf{c}, \tag{26d}$$

$$b_2 = -\boldsymbol{\omega}^T \mathbf{c}^*. \tag{26e}$$

$\because \mathbf{c}^* \notin ch(X), \therefore \mathbf{c}^* \notin X, \therefore \mathbf{x}^* \neq \mathbf{c}^*, \therefore \|\mathbf{x}^* - \mathbf{c}^*\|_2 > 0$. Then, we have
(a1) $\|\boldsymbol{\omega}\|_2 = 1$,

(a2)

$$\begin{aligned} b_2 - b_1 &= -\boldsymbol{\omega}^T \mathbf{c}^* + \boldsymbol{\omega}^T \mathbf{c} \\ &= \boldsymbol{\omega}^T \left(\mathbf{c} - \mathbf{c}^* \right) \\ &= \boldsymbol{\omega}^T \left(\frac{\mathbf{x}^* + \mathbf{c}^*}{2} - \mathbf{c} \right) \\ &= \left(\frac{\mathbf{x}^* - \mathbf{c}}{\|\mathbf{x}^* - \mathbf{c}^*\|_2} \right)^T \left(\frac{\mathbf{x}^* - \mathbf{c}^*}{2} \right) \\ &= \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2} \end{aligned} .$$

$\forall \mathbf{x} \in X, \because X \subseteq ch(X), \therefore \mathbf{x} \in ch(X)$. At the same time, according to formula (25), $\mathbf{x} \in ball\left(\mathbf{c}^*, r^*\right)$. And then we have:
(a3) the distance between $\mathbf{x}$ and hyperplane $hp\left(\boldsymbol{\omega}, b_1\right)$ is

$$\begin{aligned} dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) &= \frac{\left|\boldsymbol{\omega}^T \mathbf{x} + b_1\right|}{\|\boldsymbol{\omega}\|_2} \\ &= \frac{\boldsymbol{\omega}^T \mathbf{x} + b_1}{\|\boldsymbol{\omega}\|_2} \qquad // \ according\ to\ \textbf{Lemma}\ A.4 \\ &= \boldsymbol{\omega}^T \mathbf{x} + b_1, \qquad // \ \because (a1) \end{aligned}$$

(a4) the distance between $\mathbf{x}$ to hyperplane $hp\left(\boldsymbol{\omega}, b_2\right)$ is

$$\begin{aligned} dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_2\right)\right) &= \frac{\left|\boldsymbol{\omega}^T \mathbf{x} + b_b\right|}{\|\boldsymbol{\omega}\|_2} \\ &= \frac{\boldsymbol{\omega}^T \mathbf{x} + b_2}{\|\boldsymbol{\omega}\|_2} \qquad // \ according\ to\ \textbf{Lemma}\ A.4 \\ &= \boldsymbol{\omega}^T \mathbf{x} + b_2 \qquad // \because (a1) \\ &= \boldsymbol{\omega}^T \mathbf{x} + b_1 + \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2} \qquad // \ \because (a2) \\ &= dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) + \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2}, \qquad // \ \because (a3) \end{aligned}$$

(a5) the projection of $\mathbf{x}$ on hyperplane $hp\left(\boldsymbol{\omega}, b_1\right)$ is

$$\begin{aligned} pro\left(\mathbf{x}, hp\left(\boldsymbol{\omega}, b_1\right)\right) &= \mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|_2} \\ &= \mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) \boldsymbol{\omega}, \qquad // \ \because (a1) \end{aligned}$$

(a6) the projection of $\mathbf{x}$ on hyperplane $hp\left(\boldsymbol{\omega}, b_2\right)$ is

$$\begin{aligned} pro\left(\mathbf{x}, hp\left(\boldsymbol{\omega}, b_2\right)\right) &= \mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_2\right)\right) \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|_2} \\ &= \mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_2\right)\right) \boldsymbol{\omega} \qquad // \ \because (a1) \\ &= \mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) \boldsymbol{\omega} - \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2} \boldsymbol{\omega}, \qquad // \ \because (a5) \end{aligned}$$

(a7)

$$\begin{aligned} &\left(pro\left(\mathbf{x}, hp\left(\boldsymbol{\omega}, b_1\right)\right) - \mathbf{c}\right) - \left(pro\left(\mathbf{x}, hp\left(\boldsymbol{\omega}, b_2\right)\right) - \mathbf{c}^*\right) \\ &= \left(\mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) \boldsymbol{\omega} - \mathbf{c}\right) - \left(\mathbf{x} - dis\left(\mathbf{x}, ph\left(\boldsymbol{\omega}, b_1\right)\right) \boldsymbol{\omega} - \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2} \boldsymbol{\omega} - \mathbf{c}^*\right) \quad // \ \because (a5)\ and\ (a6) \\ &= \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2} \boldsymbol{\omega} - \mathbf{c} + \mathbf{c}^* \\ &= \frac{\|\mathbf{x}^* - \mathbf{c}^*\|_2}{2} \frac{\mathbf{x}^* - \mathbf{c}^*}{\|\mathbf{x}^* - \mathbf{c}^*\|_2} - \frac{\mathbf{x}^* + \mathbf{c}^*}{2} + \mathbf{c}^* \\ &= \frac{\mathbf{x}^* - \mathbf{c}^*}{2} - \frac{\mathbf{x}^* + \mathbf{c}^*}{2} + \mathbf{c}^* \\ &= \mathbf{0}, \end{aligned}$$

(a8)

$$\begin{aligned}
\|\mathbf{x}-\mathbf{c}\|_2^2 &= \|\mathbf{x}-pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_1\right)\right)\|_2^2+\|pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_1\right)\right)-\mathbf{c}\|_2^2 \\
&= dis^2\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_1\right)\right)+\|pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_1\right)\right)-\mathbf{c}\|_2^2 && //\ \because (a5) \\
&< \left(dis\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_1\right)\right)+\tfrac{\|\mathbf{x}^*-\mathbf{c}^*\|_2}{2}\right)^2+\|pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_2\right)\right)-\mathbf{c}^*\|_2^2 && //\ \because \|\mathbf{x}^*-\mathbf{c}^*\|_2>0\ and\ (a7) \\
&= dis^2\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_2\right)\right)+\|pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_2\right)\right)-\mathbf{c}^*\|_2^2 && //\ \because (a4) \\
&= \|\mathbf{x}-pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_2\right)\right)\|_2^2+\|pro\left(\mathbf{x},ph\left(\boldsymbol{\omega},b_2\right)\right)-\mathbf{c}^*\|_2^2 \\
&= \|\mathbf{x}-\mathbf{c}^*\|_2^2 \\
&\leq \left(r^*\right)^2. && //\ see\ \textbf{Definition}\ A.6
\end{aligned}$$

That is to say, $\exists \mathbf{c}\neq\mathbf{c}^*$, such that $\forall \mathbf{x}\in X\subseteq ch(X)$, $\|\mathbf{x}-\mathbf{c}\|_2<r^*$, which conflicts with formula (25). So the assumption is false, and then we have $\mathbf{c}^*\in ch(X)$. □

Step 4: Based on the above conclusion, the proof of **Theorem** 3.2 is given as follows.

Proof. Given a set $X\subset\mathbb{R}^d$,

$$\begin{aligned}
(C(X),R(X)) &= \underset{\mathbf{c}\in\mathbb{R}^d,\ r\in\mathbb{R}}{arg\ inf}\ r^2 && s.t.\ \forall\mathbf{x}\in X,\mathbf{x}\in ball(\mathbf{c},r) && //\ see\ \textbf{Definition}A.7 \\
&= \underset{\mathbf{c}\in\mathbb{R}^d,\ r\in\mathbb{R}}{arg\ inf}\ r^2 && s.t.\ \forall\mathbf{x}\in X,\|\mathbf{x}-\mathbf{c}\|_2^2\leq r^2 && //\ see\ \textbf{Definition}\ A.6 \\
&= \underset{\mathbf{c}\in ch(X),\ r\in\mathbb{R}}{arg\ inf}\ r^2 && s.t.\ \forall\mathbf{x}\in X,\|\mathbf{x}-\mathbf{c}\|_2^2\leq r^2 && //\ according\ to\ \textbf{Lemma}\ A.8.
\end{aligned} \tag{27}$$

Based on formula (27) and **Definition** A.1, let

$$\boldsymbol{\theta}^*=\underset{\boldsymbol{\theta}\in\triangle^{|X|}}{arg\ inf}\ g(\boldsymbol{\theta})\ \ s.t.\ \ g(\boldsymbol{\theta})=\underset{\mathbf{x}\in X}{sup}\ \|\mathbf{x}-\mathbf{X}\boldsymbol{\theta}\|_2^2, \tag{28}$$

then, $C(X)=\mathbf{X}\boldsymbol{\theta}^*$ and $R^2(X)=g(\boldsymbol{\theta}^*)=\underset{\mathbf{x}\in X}{sup}\ \|\mathbf{x}-\mathbf{X}\boldsymbol{\theta}^*\|_2^2$. That is to say,
(a1) optimization problem (27) can be solved equivalently by solving optimization problem (28).

(a2) $\forall\mathbf{x}_i\in X$,

$$g_i(\boldsymbol{\theta})=\|\mathbf{x}_i-\mathbf{X}\boldsymbol{\theta}\|_2^2=\boldsymbol{\theta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\theta}-2\mathbf{x}_i^T\mathbf{X}\boldsymbol{\theta}+\mathbf{x}_i^T\mathbf{x}_i$$

is a quadratic function w.r.t $\boldsymbol{\theta}$. In $g_i(\boldsymbol{\theta})$, $\because \mathbf{X}^T\mathbf{X}$ is a symmetric positive semidefinite matrix, $\therefore g_i(\boldsymbol{\theta})$ is a convex function w.r.t $\boldsymbol{\theta}$.

Because point-wise supremum over an infinite set is a operation that preserve convexity (a formal description can been found in Page 81 of (Boyd & Vandenberghe, 2004)), combining (a2), we have
(a3) given X,

$$g(\boldsymbol{\theta})=\underset{\mathbf{x}_i\in X}{sup}\ g_i\left(\boldsymbol{\theta}\right)=\underset{\mathbf{x}\in X}{sup}\ \|\mathbf{x}-\mathbf{X}\boldsymbol{\theta}\|_2^2$$

is also a convex function w.r.t $\boldsymbol{\theta}$.

At the same time, (a4) $\triangle^{|X|}$ only involves linear constraints.

Combining (a3) and (a4), we have (a5) formula (28) is a convex optimization problem.

Combining (a1) and (a5), the theorem is proofed. □

A.3. Proof of Formula (11)

Proof. Given a set of positive samples $X_+=\left\{\mathbf{x}_i|\mathbf{x}_i\in\mathbb{R}^d,i=1,2,\cdots,n_+\right\}$, a set of negative samples $X_-=\left\{\mathbf{x}_j|\mathbf{x}_j\in\mathbb{R}^d,j=n_++1,n_++2,\cdots,n_++n_-\right\}$, $n=n_-+n_+$, and a kernel function $\varphi:\mathbb{R}^d\to\mathbb{R}^{d'}$.

Let

$$\varphi(\mathbf{X}_+) = \left(\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \cdots \varphi\left(\mathbf{x}_{n_+}\right)\right) \in \mathbb{R}^{d' \times n_+}, \tag{29a}$$

$$\varphi(\mathbf{X}_-) = \left(\varphi\left(\mathbf{x}_{n_++1}\right), \varphi\left(\mathbf{x}_{n_++2}\right), \cdots \varphi\left(\mathbf{x}_{n_++n_-}\right)\right) \in \mathbb{R}^{d' \times n_-}, \tag{29b}$$

$$\varphi(\mathbf{X}) = \left(\varphi(\mathbf{X}_+), \varphi(\mathbf{X}_-)\right) \in \mathbb{R}^{d' \times n}, \tag{29c}$$

$$\mathbf{K}^{\varphi} = \varphi(\mathbf{X})^T \varphi(\mathbf{X}) \in \mathbb{R}^{n \times n}, \tag{29d}$$

$$\hat{\mathbf{K}}^{\varphi} = \begin{pmatrix} \varphi(\mathbf{X}_+)^T \varphi(\mathbf{X}_+), & -\varphi(\mathbf{X}_+)^T \varphi(\mathbf{X}_-) \\ -\varphi(\mathbf{X}_-)^T \varphi(\mathbf{X}_+), & \varphi(\mathbf{X}_-)^T \varphi(\mathbf{X}_-) \end{pmatrix} \in \mathbb{R}^{n \times n}. \tag{29e}$$

Obviously, matrices defined in formula (29d) and (29e) are both symmetric positive semidefinite.

(a1) For radius problem,

$$\begin{aligned} g_1(\varphi) &= \min_{\mathbf{u}} \max_{\mathbf{x}_i \in X} \|\varphi(\mathbf{x}_i) - \varphi(\mathbf{X})\sigma(\mathbf{u})\|_2^2 && //see\ formula\ (10) \\ &= \min_{\mathbf{u}} \max_{\mathbf{x}_i \in X} \sigma(\mathbf{u})^T \varphi(\mathbf{X})^T \varphi(\mathbf{X}) \sigma(\mathbf{u}) - 2\varphi(\mathbf{x}_i)^T \varphi(\mathbf{X}) \sigma(\mathbf{u}) \\ &\qquad + \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_i) \\ &= \min_{\mathbf{u}} \max_{\mathbf{x}_i \in X} \sigma(\mathbf{u})^T \mathbf{K}^{\varphi} \sigma(\mathbf{u}) - 2\mathbf{K}^{\varphi}_{[i,:]} \sigma(\mathbf{u}) + k^{\varphi}_{ii}. && //\ see\ formula\ (29d) \end{aligned} \tag{30}$$

where $\mathbf{K}^{\varphi}_{[i,:]}$ is the i-th row of matrix $\mathbf{K}^{\varphi}$.

(a2) For margin problem,

$$\begin{aligned} g_2(\varphi) &= \min_{\mathbf{v},\mathbf{w}} \|\varphi(\mathbf{X}_+)\sigma(\mathbf{v}) - \varphi(\mathbf{X}_-)\sigma(\mathbf{w})\|_2^2 && //see\ formula\ (10) \\ &= \min_{\mathbf{v},\mathbf{w}} \sigma(\mathbf{v})^T \varphi(\mathbf{X}_+)^T \varphi(\mathbf{X}_+) \sigma(\mathbf{v}) - 2\sigma(\mathbf{v})^T \varphi(\mathbf{X}_+)^T \varphi(\mathbf{X}_-) \sigma(\mathbf{w}) \\ &\qquad + \sigma(\mathbf{w})^T \varphi(\mathbf{X}_-)^T \varphi(\mathbf{X}_-) \sigma(\mathbf{w}) \\ &= \min_{\mathbf{v},\mathbf{w}} \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}^T \begin{pmatrix} \varphi(\mathbf{X}_+)^T \varphi(\mathbf{X}_+), & -\varphi(\mathbf{X}_+)^T \varphi(\mathbf{X}_-) \\ -\varphi(\mathbf{X}_-)^T \varphi(\mathbf{X}_+), & \varphi(\mathbf{X}_-)^T \varphi(\mathbf{X}_-) \end{pmatrix} \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix} \\ &= \min_{\mathbf{v},\mathbf{w}} \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}^T \hat{\mathbf{K}}^{\varphi} \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}. && //\ see\ formula\ (29e) \end{aligned} \tag{31}$$

Combining (a1) and (a2), the formula (11) is proofed. □

B. Algorithms

B.1. Algorithms for Solving Formula (10)

To solve the optimization problems $g_1(\varphi)$ and $g_2(\varphi)$ in formula (10), the **Algorithm** 1 and 2 are given as follows.

B.1.1. Convergence Analysis

In **Algorithm** 1 and 2, the initial solution and solutions obtained at each iteration are both feasible solutions, because the output of the softmax activation function $\sigma(\cdot)$ is the convex combination coefficients. So **Algorithm** 1 and 2 are both interior point methods and the convergence of them can be guaranteed (Boyd & Vandenberghe, 2004).

B.1.2. Efficiency Analysis

The efficiency of **Algorithm** 1 and 2 can be analyzed from the following three aspects. First, the constant vector $\sigma((1, 1, \cdots, 1)^T)$ with adaptive length is used as the initial point, which is a strictly feasible solution. The computational complexity of calculating the initial point is of constant order. Second, the accelerated first-order algorithm, e.g., Adam (Kingma & Ba, 2015) can be used to update the optimization variables. Third, these algorithms can easily leverage high-performance computing hardware, e.g., GPU for acceleration. In this paper, we utilize the NVIDIA GeForce RTX 3090 for acceleration.

Algorithm 1 Solving $g_1(\varphi)$ in Formula (10)

Input: The set of training samples $X = \{\mathbf{x}_i\}_{i=1}^n$, the representation learning function φ, the maximum number of iterations T, the error tolerance parameter ϵ.

Output: The optimal value g^* and optimal solution $\mathbf{u}^*$.

Initialize $\mathbf{u} \leftarrow (1, 1, \cdots, 1)^T \in \mathbb{R}^n$, $t \leftarrow 0$, $err \leftarrow +\infty$.
if φ is kernel function **then**
  Compute kernel matrix $\mathbf{K}^{\varphi}$ by formula (29d).
else
  Compute latent representation matrix $\varphi(\mathbf{X}) \leftarrow (\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \cdots, \varphi(\mathbf{x}_n))$.
end if
if φ is kernel function **then**
  $g_{\text{new}} \leftarrow \underset{\mathbf{x}_i \in X}{max} \ \sigma(\mathbf{u})^T \mathbf{K}^{\varphi} \sigma(\mathbf{u}) - 2\mathbf{K}^{\varphi}_{[i,:]} \sigma(\mathbf{u}) + k^{\varphi}_{ii}$.
else
  $g_{\text{new}} \leftarrow \underset{\mathbf{x}_i \in X}{max} \ \|\varphi(\mathbf{x}_i) - \varphi(\mathbf{X}) \sigma(\mathbf{u})\|_2^2$.
end if
while $t < T$ **and** $err > \epsilon$ **do**
  $g_{\text{old}} \leftarrow g_{\text{new}}$.
  Update $\mathbf{u}$ by Adam (Kingma & Ba, 2015) with loss g_{old}.
  if φ is kernel function **then**
    $g_{\text{new}} \leftarrow \underset{\mathbf{x}_i \in X}{max} \ \sigma(\mathbf{u})^T \mathbf{K}^{\varphi} \sigma(\mathbf{u}) - 2\mathbf{K}^{\varphi}_{[i,:]} \sigma(\mathbf{u}) + k^{\varphi}_{ii}$
  else
    $g_{\text{new}} \leftarrow \underset{\mathbf{x}_i \in X}{max} \ \|\varphi(\mathbf{x}_i) - \varphi(\mathbf{X}) \sigma(\mathbf{u})\|_2^2$.
  end if
  $t \leftarrow t + 1$.
  $err \leftarrow |g_{\text{old}} - g_{\text{new}}|$.
end while
return g_{new} and $\mathbf{u}$.

Algorithm 2 Solving $g_2(\varphi)$ in Formula (10)

Input: The set of positive training samples $X_+ = \{\mathbf{x}_i\}_{i=1}^{n_+}$, the set of negative training samples $X_- = \{\mathbf{x}_j\}_{j=n_++1}^{n_++n_-}$, the representation learning function φ, the maximum number of iterations T, the error tolerance parameter ϵ.

Output: The optimal value g^* and optimal solution $(\mathbf{v}^*, \mathbf{w}^*)$.

Initialize $\mathbf{v} \leftarrow (1, 1, \cdots, 1)^T \in \mathbb{R}^{n_+}$, $\mathbf{w} \leftarrow (1, 1, \cdots, 1)^T \in \mathbb{R}^{n_-}$, $t \leftarrow 0$, $err \leftarrow +\infty$.
if φ is kernel function **then**
  Compute kernel matrix $\hat{\mathbf{K}}^{\varphi}$ by formula (29e).
else
  $\varphi(\mathbf{X}_+) \leftarrow \left(\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \cdots, \varphi\left(\mathbf{x}_{n_+}\right)\right)$, $\varphi(\mathbf{X}_-) \leftarrow \left(\varphi\left(\mathbf{x}_{n_++1}\right), \varphi\left(\mathbf{x}_{n_++2}\right), \cdots, \varphi\left(\mathbf{x}_{n_++n_-}\right)\right)$.
end if
if φ is kernel function **then**
  $g_{\text{new}} \leftarrow \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}^T \hat{\mathbf{K}}^{\varphi} \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}$
else
  $g_{\text{new}} \leftarrow \|\varphi(\mathbf{X}_+)\sigma(\mathbf{v}) - \varphi(\mathbf{X}_-)\sigma(\mathbf{w})\|_2^2$.
end if
while $t < T$ **and** $err > \epsilon$ **do**
  $g_{\text{old}} \leftarrow g_{\text{new}}$.
  Update $\mathbf{v}$ and $\mathbf{w}$ by Adam (Kingma & Ba, 2015) with loss g_{old}.
  if φ is kernel function **then**
    $g_{\text{new}} \leftarrow \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}^T \hat{\mathbf{K}}^{\varphi} \begin{pmatrix} \sigma(\mathbf{v}) \\ \sigma(\mathbf{w}) \end{pmatrix}$
  else
    $g_{\text{new}} \leftarrow \|\varphi(\mathbf{X}_+)\sigma(\mathbf{v}) - \varphi(\mathbf{X}_-)\sigma(\mathbf{w})\|_2^2$.
  end if
  $t \leftarrow t + 1$.
  $err \leftarrow |g_{\text{old}} - g_{\text{new}}|$.
end while
return g_{new} and $(\mathbf{v}, \mathbf{w})$.

B.2. Kernel Selection Algorithm

The detailed description of VC dimension based kernel selection method is given in **Algorithm** 3.

Algorithm 3 VC Dimension based Kernel Selection

Input: The set of positive training samples $X_+ = \{\mathbf{x}_i\}_{i=1}^{n_+}$, the set of negative training samples $X_- = \{\mathbf{x}_j\}_{j=n_++1}^{n_++n_-}$, the set of candidate kernel functions $\{\varphi_k\}_{k=1}^{m}$.
Output: The optimal kernel function φ_{opt}.

Initialize $f_{min} \leftarrow +\infty$, $\varphi_{opt} \leftarrow$ **None**.
for $k = 1$ **to** m **do**
  $\varphi \leftarrow \varphi_k$.
  Solve optimization problem $g_1(\varphi)$ in formula (11) on $X_+ \bigcup X_-$ by **Algorithm** 1.
  Solve optimization problem $g_2(\varphi)$ in formula (11) on X_+ and X_- by **Algorithm** 2.
  $f(\varphi) \leftarrow 4\frac{g_1(\varphi)}{g_2(\varphi)}$.
  if $f(\varphi) < f_{min}$ **then**
    $f_{min} \leftarrow f(\varphi)$.
    $\varphi_{opt} \leftarrow \varphi$.
  end if
end for
return φ_{opt}.

B.2.1. Efficiency Analysis

Cross validation is one of the most popular methods for kernel selection. We use SVM-based k-fold cross verification as the benchmark to analyze the efficiency of the proposed kernel selection method.

Given n training samples and a kernel function φ.

To evaluate the quality of φ, the proposed method needs to solve two constrained convex quadratic programming problems, i.e., $g_1(\varphi)$, $g_2(\varphi)$ in formula (9). In each optimization problem, the scale of the optimization variable is $O(n)$ and the scale of the constraint condition is $O(n)$. Let $O(T)$ denote the computational cost of solving the optimization problem $g_1(\varphi)$ (or $g_2(\varphi)$), and then the computational complexity of the proposed method is $O(2T) = O(T)$.

To evaluate the quality of φ, SVM-based k-fold cross verification method needs to solve k convex quadratic programming problems corresponding to SVM, denoted as $g_{SVM}(\varphi)$ [1]. In each $g_{SVM}(\varphi)$, the scale of the optimization variable is $O(\frac{k-1}{k}n) = O(n)$ and the scale of the constraint condition is $O(\frac{k-1}{k}n) = O(n)$. Therefore, the computational cost of solving $g_{SVM}(\varphi)$ is $O(T)$, which is same as $g_1(\varphi)$. In summary, the computational complexity of k-fold cross-verification is $O(kT)$.

In general, we have $k > 2$, so the computational complexity of the proposed method has certain advantages compared with SVM-based k-fold cross verification method.

Thanks to the good property of optimization problem (9), the constraint conditions in formula (9) can be converted into a softmax activation function layer. This allows some acceleration methods to be directly used. Specifically, (a) the accelerated first-order algorithm, e.g., Adam (Kingma & Ba, 2015) is used for solving, (b) the mature deep learning architecture, e.g., Pytorch [2] is used for programming, (c) the high-performance computing hardware, e.g., GPU is used for acceleration. As a result, the running time of the proposed method has great advantages compared with the SVM-based k-fold cross verification method.

B.3. DNN Boosting Framework Algorithm

The detailed description of VC dimension based DNN boosting framework is given in **Algorithm** 4.

[1] The computational complexity of the test process is ignored.
[2] https://pytorch.org/

Algorithm 4 VC Dimension based DNN Boosting Framework

Input: The training data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, output space $\mathcal{Y}$, the representation learning network $\varphi(\cdot;\Theta)$, the 2-norm standardization network $\lambda(\cdot)$, the classification network $h(\cdot;\mathbf{W},\mathbf{b})$, the weight of VC dimension based loss γ_{VC}, the number of iterations T, the size of batch s.

Output: The final model.

1: Initialize Θ, $\mathbf{W}$ and $\mathbf{b}$ randomly.
2: **for** $t = 1$ **to** T **do**
3: $D' \leftarrow$ select s samples from training data set D randomly, $\mathcal{Y}' \leftarrow \{y_i | (\mathbf{x}_i, y_i) \in D'\}$, such that $|\mathcal{Y}'| \geq 2$.
4: $\mathcal{L}_{VC} \leftarrow 0$, $k \leftarrow 0$.
5: **for** $p, q \in \mathcal{Y}', p < q$ **do**
6: Compute set of positive samples $X_+ \leftarrow \{x_i | (x_i, y_i) \in D', y_i = p\}$.
7: Compute set of negative samples $X_- \leftarrow \{x_i | (x_i, y_i) \in D', y_i = q\}$.
8: Compute $l_{pq} \leftarrow -g^{(p,q)}(\lambda \circ \varphi)$ in formula (14) on X_+ and X_- by **Algorithm** 2.
9: $\mathcal{L}_{\mathrm{VC}} \leftarrow \mathcal{L}_{\mathrm{VC}} + l_{pq}$, $k \leftarrow k + 1$.
10: **end for**
11: Compute empirical loss $\mathcal{L}$ in formula (14) on data set D'.
12: Update Θ, $\mathbf{W}$, and $\mathbf{b}$ by Adam (Kingma & Ba, 2015) with loss $\mathcal{L} + \frac{\gamma_{\mathrm{VC}}}{k}\mathcal{L}_{\mathrm{VC}}$.
13: **end for**
14: **return** $h(\lambda(\varphi(\cdot;\Theta));\mathbf{W},\mathbf{b})$.

B.3.1. Efficiency Analysis

Stochastic gradient descent with mini-batch is a popular optimizing strategy in deep learning. There are many accelerated first-order algorithms, e.g., Adam (Kingma & Ba, 2015) for training DNN effectively and efficiently. Meanwhile, with the help of mature deep learning frameworks e.g., Pytorch, the high-performance computing hardware e.g., GPU can be used for acceleration. These strategies can be directly applied to the proposed framework to achieve efficient training.

When the number of the training samples n is large, we can randomly sample a min-batch of training samples D' such that $|D'| \ll n$. When the number of classes $|\mathcal{Y}|$ is large, we can randomly sample a min-batch of training samples D' such that $2 \leq |\{y_i | (\mathbf{x}_i, y_i) \in D'\}| \ll |\mathcal{Y}|$. And then the VC dimension based loss $\mathcal{L}_{\mathrm{VC}}$ (lines 4-10 of **Algorithm** B.3) and the empirical loss $\mathcal{L}$ (line 11 of **Algorithm** B.3) can be calculated efficiently. Based on the loss, the parameters of the model can be updated efficiently (line 12 of **Algorithm** B.3).

C. Experimental Details

C.1. Details of Verifying Theoretical Results

Data set For Taichi date set, input space is $\mathcal{X} = \left\{(x_1, x_2)^T | x_1^2 + x_2^2 \leq 4^2\right\}$ and samples obey the uniform distribution on $\mathcal{X}$. Output space is $\mathcal{Y} = \{+1, -1\}$. Target function f_{c} is nonlinear and is defined as follows

$$\forall (x_1, x_2)^T \in \mathcal{X}, f_{\mathrm{c}}\left((x_1, x_2)^T\right) = \begin{cases} +1, & x_1^2 + (x_2 - 2)^2 < 1^2 \\ -1, & x_1^2 + (x_2 + 2)^2 < 1^2 \\ g_{\mathrm{c}}\left((x_1, x_2)^T\right), & otherwise \end{cases},$$

where

$$g_{\mathrm{c}}\left((x_1, x_2)^T\right) = \begin{cases} +1, & x_1 \geq 0, x_2 \geq 0, x_1^2 + (x_2 - 2)^2 > 2^2 \\ -1, & x_1 \geq 0, x_2 \geq 0, x_1^2 + (x_2 - 2)^2 \leq 2^2 \\ -1, & x_1 < 0, x_2 \geq 0 \\ -1, & x_1 < 0, x_2 < 0, x_1^2 + (x_2 + 2)^2 > 2^2 \\ +1, & x_1 < 0, x_2 < 0, x_1^2 + (x_2 + 2)^2 \leq 2^2 \\ +1, & x_1 \geq 0, x_2 < 0 \end{cases}.$$

Code implementations and parameter settings The basic learner SVM is implemented by Scikit-learn [3]. All parameters adopt the default settings except for kernel function. For the proposed method, **Algorithm** 1 and 2 are called to calculate the objective function $f\left(\varphi(\cdot;\delta)\right)$ in formula (10). In the whole experiment, $T = 300$, $\epsilon = 10^{-10}$ are adopted for **Algorithm** 1 and 2. The Adam (Kingma & Ba, 2015) used in **Algorithm** 1 and 2 is implemented by Pytorch, and all parameters adopt the default settings.

C.2. Details of Kernel Selection Experiment

Data sets There are 15 UCI [4] binary classification data sets and their basic information is given in **Table** 5.

Table 5. Basic information of 15 binary classification data sets.

ID	DATA SET	SAMPLES	FEATURES	CLASS RATIO
D1	TRAINS	10	29	1.00
D2	SPECT	79	22	2.04
D3	MOLEC-BIOL-PROMOTER	106	57	1.00
D4	MONKS-2	169	6	1.64
D5	PLANNING	182	12	2.50
D6	PARKINSONS	195	22	3.06
D7	HEART-HUNGARIAN	294	12	1.77
D8	HABERMAN-SURVIVAL	306	3	2.78
D9	BREAST-CANCER-WISC-DIAG	569	30	1.68
D10	ILPD-INDIAN-LIVER	583	9	2.49
D11	HILL-VALLEY	606	100	1.03
D12	CREDIT-APPROVAL	690	15	1.25
D13	BREAST-CANCER-WISC	699	9	1.90
D14	RINGNORM	7,400	20	1.02
D15	MAGIC	19,020	10	1.84

Code implementations and parameter settings The settings of basic learner SVM and the proposed method are the same as that in **Appendix** C.1. The experiments of 5-fold cross validation based on the SVM implemented by Scikit-learn are conducted on Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz. The experiments of the proposed method implemented based on Pytorch are conducted on NVIDIA GeForce RTX 3090.

C.3. Details of DNN Boosting Experiment

Data sets The basic information of two image classification data sets is given as follows.

MNIST [5] is a handwritten digits classification data set, which consists of 10 classes, i.e., $\{0, 1, \cdots, 9\}$. It has 60,000 training samples and 10,000 test samples. And each sample is an image with 28 $\times$ 28 pixels.

CIFAR10 [6] is a visual objects classification data set, which consists of 10 classes, i.e., {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}. It has 50,000 training samples and 10,000 test samples. And each sample is an image with 3 $\times$ 32 $\times$ 32 pixels.

Basic DNNs The brief introductions of 5 basic DNNs are given as follows.

FCNet3: A 3-layers fully connected network. The representation learning network is

$$Linear(d, 1024) - ReLU() - Linear(1024, 512) - RelU() - Linear(512, d'),$$

and classification network is

$$Linear(d', |\mathcal{Y}|),$$

[3] https://scikit-learn.org/stable/
[4] https://archive.ics.uci.edu/ml/index.php
[5] http://yann.lecun.com/exdb/mnist/
[6] http://www.cs.toronto.edu/~kriz/cifar.html

where for MNIST data set, $d = 28 \times 28 = 784$, for CIFAR10 data set $d = 3 \times 32 \times 32 = 3072$, and for MNIST and CIFAR10 data sets $d' = 256$, $|\mathcal{Y}| = 10$.

LeNet (LeCun et al., 1989): A classical CNN for image classification.

ResNet18 and ResNet50 (He et al., 2016): Two commonly used CNNs with residual connections for image classification.

ViT-Base (Dosovitskiy et al., 2021): A recently proposed DNN that has achieved state-of-the-art performance in many visual tasks.

Code implementations and parameter settings The Adam (Kingma & Ba, 2015) implemented by Pytorch is used to train basic models (FCNet3, LeNet, ResNet18, ResNet50, ViT-Base), the models embedded into DeepLDA (Dorfer et al., 2016) (FCNet3+LDA, LeNet+LDA, ResNet18+LDA, ResNet50+LDA, and ViT-Base+LDA), and the models embedded into the proposed framework (FCNet3+Our, LeNet+Our, ResNet18+Our, ResNet50+Our, and ViT-Base+Our).

In the whole experiment, the set of candidate learning rate of Adam is $\{5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}\}$, and the rest parameters adopt the default settings. At the same time, the number of epoch is 500.

For the proposed method, the set of candidate trade-off parameter γ_{VC} is $\{10^{-2}, 10^{-1}, 1\}$. At the same time, **Algorithm** 2 is called to calculate the VC dimension based loss in formula (14). For **Algorithm** 2, $T = 100$ and $\epsilon = 10^{-10}$ are adopted.

D. Code Release

The codes of the proposed methods are available at `https://github.com/JunbiaoCui/GRLF_GPG`.