AIMS Mathematics, 11(8): 26699–26719.
DOI: 10.3934/math.20261071
Received: 10 June 2026
Revised: 04 August 2026
Accepted: 19 August 2026
Published: 25 August 2026

https://www.aimspress.com/journal/Math

Research article

Riemannian gradient-based Zhang neural network method for linear discriminant analysis

Yu-Hang Wang, Zhuo-Cheng Xie, Meng-Zhen Fu* and Huan Ren

School of Mathematical Sciences, Jiangxi Science and Technology Normal University, Nanchang, 330038, China

* **Correspondence:** Email: fumengzhen@jxstnu.edu.cn.

Abstract: Linear discriminant analysis (LDA) is a classic statistical method employed for classification and dimensionality reduction of large-scale datasets. In this paper, we propose a Zhang neural network method based on Riemannian gradient. Compared with traditional approaches for solving LDA, the proposed method achieves higher convergence accuracy, accelerated convergence speed, and reduced parameter sensitivity. In terms of theoretical analysis, we have investigated the properties and geometric dynamic characteristics, and provided stability analysis of the algorithm. Finally, through experiments with synthetic data and ORL face images, and by making a clear comparison with WH2, the advantages of the proposed algorithm were verified.

Keywords: linear discriminant analysis; Zhang neural network method; generalized Stiefel manifold; generalized Stiefel manifold
Mathematics Subject Classification: 15A99, 65F99

1. Introduction

Linear discriminant analysis (LDA) is a classical supervised dimensionality reduction method, first proposed by Ronald Fisher [1] in 1936 for binary classification problems. Its essence is to find a projection vector such that, after projecting the training samples onto this axis, the within-class scatter matrix of the training dataset is minimized while the between-class scatter matrix is simultaneously maximized.

Based on Fisher's pioneering research, Duda [2, 3] formalized the notion of discriminant vector sets. The core idea is to project raw samples onto a subspace spanned by discriminant vectors, where classification is subsequently conducted. This framework extends linear discriminant analysis (LDA) to multi-class classification scenarios. To advance LDA performance in low-dimensional settings, Kothari and Lotlikar [4] developed fractional-step linear discriminant analysis (FLDA). A well-known

obstacle of conventional LDA arises when the within-class scatter matrix becomes singular. To tackle this issue, Belhumeur et al. [5] adopted principal component analysis (PCA) to eliminate the null space of the within-class scatter matrix, converting the original LDA task into a generalized eigenvalue problem. Complementarily, Liu et al. [6,7] put forward algorithms to capture informative discriminant features residing in the null space. In contrast, Yu and Yang [8] argued that discriminant vectors lying in the null space of the between-class scatter matrix contribute little to discrimination, and accordingly proposed direct LDA (DLDA). DLDA operates directly on original high-dimensional data and discards the null space of the between-class scatter matrix. Along another research line, Loog et al. [9] modified the construction of the between-class scatter matrix by introducing a weighting function. To balance the bias and variance associated with eigenvalues of the within-class scatter matrix, Lu et al. [10] introduced tunable regularization parameters and established regularized LDA (RLDA). Further variants of LDA impose additional structural constraints on discriminant directions. Foley and Sammon proposed orthogonal LDA [11], which seeks optimal discriminant vectors orthogonal with respect to the total scatter matrix. Following this line, Jin et al. proposed uncorrelated LDA [12], whose extracted low-dimensional feature vectors are mutually uncorrelated and therefore reduce redundant information. Later, Chu [13] presented regularized orthogonal linear discriminant analysis (ROLDA), equipped with mathematical criteria to select proper regularization parameters.

The aforementioned methods are mainly defined within Euclidean space. In recent years, manifold learning has emerged as a powerful paradigm to exploit intrinsic geometric structures hidden in high-dimensional data [14, 15]. Moving beyond this setting, Zhou [16] proposed manifold partition discriminant analysis (MPDA), which integrates discriminant learning with intrinsic data manifold structures and achieves favorable classification results for datasets with complex distributions. Yao et al. designed an adaptive-weighted ensemble CNN-LSTM-GRU prediction framework optimized by an improved sparrow search algorithm, where low-dimensional discriminant meteorological and runoff features extracted by the optimized deep model significantly boost long-term hydrological prediction accuracy [17].

In recent years, the Zhang neural network (ZNN), a class of recurrent neural dynamics, has attracted extensive attention for addressing matrix numerical computation tasks. Originally proposed by Zhang [18], ZNN is essentially a gradient-based zeroing neural network, which differs fundamentally from classical deep learning neural networks. Uhlig [19] systematically investigated the theoretical properties of ZNN and summarized a standardized seven-step framework for constructing ZNN solvers. Benefiting from inherent parallelizability, ZNN is capable of handling both time-varying and time-invariant computational problems efficiently. A variety of matrix-related tasks have been successfully solved within the ZNN framework. For example, Wei et al. developed ZNN approaches for generalized eigenvalue problems [20] and total least squares problems [21]. Weingessel and Hornik [22] constructed a neural network to compute dominant singular triplets without performing covariance matrix deflation required by APEX-type algorithms. Huang and Li [23] further extended neural dynamic methods to calculate the quaternion rank-k truncated singular value decomposition. Moreover, ZNN techniques have been developed to deal with matrix equations [24–26], matrix pseudoinversion [27], and matrix optimization problems [28]. Most of the above studies are established in Euclidean space. To cope with optimization tasks with manifold constraints, recent attempts combine ZNN dynamics with Riemannian optimization. Notably, Xie et al. [29] constructed a Riemannian gradient-based ZNN to solve canonical correlation analysis on generalized Stiefel

manifolds, demonstrating the feasibility of integrating ZNN with manifold geometry. Nevertheless, although ZNNs have been utilized to tackle principal component analysis and various matrix equation problems, a specialized Riemannian gradient-driven ZNN designed for LDA subject to generalized Stiefel manifold constraints remains absent, thereby inspiring this research.

Although the WH2 algorithm is applicable for solving the LDA problem, it has evident limitations, including unsatisfactory convergence precision, slow convergence rate, and pronounced sensitivity to the sampling-time parameter τ. Inspired by the work of solving canonical correlation analysis by the Zhang neural network method, this paper proposes a Riemannian gradient-based Zhang neural network method for linear discriminant analysis. Compared with WH2, the proposed method achieves higher convergence accuracy, accelerated convergence speed, and reduced parameter sensitivity. This novel iterative method first discretizes the Riemannian gradient of the optimization problem with generalized orthogonality constraints, and then iteratively drives the Riemannian gradient to zero efficiently to derive the optimal solution. Theoretically, the decay rate of the gradient in the neural network method is exponential. Furthermore, we prove that the iterative sequence generated by the neural network method maintains generalized orthogonality. Finally, synthetic and real data experiments demonstrate the validity and practical utility of this ZNN method.

The rest of this paper is organized as follows. In Section 2, we briefly review the definition of LDA and the properties of Riemannian gradient. In Section 3, we propose a neural network method for LDA based on Riemannian gradient. In Section 4, we prove that the iterative sequence generated by the proposed neural network method consistently maintains generalized orthogonality, and provide a stability analysis of the method. In Section 5, we validate the effectiveness of the proposed method through some synthetic and real data experiments. Some conclusions and remarks are presented in Section 6.

2. Preliminaries

2.1. LDA

Given a data matrix $A = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n] \in \mathbb{R}^{m\times n}$ consisting of n training samples in an m-dimensional feature space, we artificially partition A into c classes as $A = [A_1, A_2, ..., A_c]$. Here, A_i denotes the i-th class subset containing n_i samples, and the class sizes satisfy $n_1 + n_2 + \cdots + n_c = n$. Denote by $\bar{\mathbf{a}}$ the overall mean of all training samples, by $\bar{\mathbf{a}}_i$ the mean of training samples in the i-th class, and by $\mathbf{a}_j^{(i)}$ the j-th training sample in the i-th class. Letting $\mathbf{1}_i = [1, 1, ..., 1]^T \in \mathbb{R}^{n_i}$, the within-class scatter matrix S_w and the between-class scatter matrix S_b are defined as follows:

$$\begin{aligned} S_w &= \sum_{i=1}^{c}\sum_{j=1}^{n_i}(\mathbf{a}_j^{(i)} - \bar{\mathbf{a}}_i)(\mathbf{a}_j^{(i)} - \bar{\mathbf{a}}_i)^T = P_w {P_w}^T, \\ S_b &= \sum_{i=1}^{c} n_i(\bar{\mathbf{a}}_i - \bar{\mathbf{a}})(\bar{\mathbf{a}}_i - \bar{\mathbf{a}})^T = P_b {P_b}^T, \end{aligned} \tag{2.1}$$

where $P_w = [A_1 - \bar{\mathbf{a}}_i\mathbf{1}_i^T, ..., A_c - \bar{\mathbf{a}}_c\mathbf{1}_c^T] \in \mathbb{R}^{m\times n}$ and $P_b = [\sqrt{n_1}(\bar{\mathbf{a}}_1 - \bar{\mathbf{a}}), \sqrt{n_2}(\bar{\mathbf{a}}_2 - \bar{\mathbf{a}}), ..., \sqrt{n_c}(\bar{\mathbf{a}}_c - \bar{\mathbf{a}})] \in \mathbb{R}^{m\times c}$. In addition, the total scatter matrix is defined as

$$S_t = \sum_{k=1}^{n}(\mathbf{a}_k - \bar{\mathbf{a}})(\mathbf{a}_k - \bar{\mathbf{a}})^T = P_t {P_t}^T,$$

where $P_t = [\mathbf{a}_1 - \bar{\mathbf{a}}, \mathbf{a}_2 - \bar{\mathbf{a}}, ..., \mathbf{a}_n - \bar{\mathbf{a}}] \in \mathbb{R}^{m\times n}$. Obviously,

$$S_t = S_w + S_b.$$

The LDA method aims to find the Fisher projection matrix Q such that the within-class scatter distance is minimized and the between-class scatter distance is maximized:

$$J(Q) = \max_{Q} \frac{\text{trace}(Q^T S_b Q)}{\text{trace}(Q^T S_w Q)}, \tag{2.2}$$

where $Q \in \mathbb{R}^{m\times r}$, and $r < m$ is the reducing dimension.

2.2. Riemannian gradient

Definition 2.1. *[30] If $X \in \mathbb{R}^{m\times l}$, $S \in \mathbb{R}^{m\times m}$ is a symmetric positive definite matrix, and $X^T S X = I$, then we call the set $\text{St}_S(m, l)$ the generalized Stiefel manifold, and denote it as*

$$\text{St}_S(m, l) = \{X \in \mathbb{R}^{m\times l} : X^T S X = I\}.$$

For the tangent space of a generalized Stiefel manifold, its definition is as follows.

Definition 2.2. *[30] The tangent space of matrix X on the generalized Stiefel manifold $\text{St}_S(m, l)$ is $T_X\text{St}_S(m, l)$, which is denoted as*

$$T_X\text{St}_S(m, l) = \{A \in \mathbb{R}^{m\times l} : X^T S A + A^T S X = 0\}.$$

As this manifold structure violates the closure of vector space under linear operations, classical differentiation theory requires geometric adaptation through Riemannian metrics. Next, based on the generalized Stiefel manifold, we endow it with a non-standard Riemannian metric.

Definition 2.3. *[30] For $X \in \text{St}_S(m, l)$ and $\forall A, B \in T_X\text{St}_S(m, l)$, we define the following Riemannian metric:*

$$\langle A, B\rangle_X = \text{trace}(B^T S A),$$

the induced norm corresponding to the X-inner product is defined as:

$$\|A\|_X = \sqrt{\langle A, A\rangle_X} = \sqrt{\text{trace}(A^T S A)}.$$

Following this theoretical framework, we first recall the tangent space of the generalized Stiefel manifold, then establish the Riemannian gradient.

Definition 2.4. *[30] For $X \in \text{St}_S(m, l)$, given a smooth function f on the generalized Stiefel manifold $\text{St}_S(m, l)$, we define the Riemannian gradient of f as* grad $f(X)$*, satisfying*

$$\langle \text{grad } f(X), \eta\rangle_X = \text{D } f(X)[\eta], \quad \forall\ \eta \in T_X\text{St}_S(m, l),$$

where D $f(X)[\eta]$ *is the directional derivative of f at the point X in a given direction η.*

Lemma 2.5. *[31] Given $X \in \text{St}_S(m, l)$ and $A \in T_X\text{St}_S(m, l)$, then the projection of A onto $T_X\text{St}_S(m, l)$ is given by*

$$\mathcal{P}_{T_X}(A) = A - X\text{sym}(X^T S A),$$

where $\text{sym}(X^T S A) = (X^T S A + A^T S^T X)/2$.

Lemma 2.6. *[30] Let the generalized Stiefel manifold $\mathcal{M}$ be an embedded submanifold of a Riemannian manifold $\overline{\mathcal{M}}$, $\bar{f}$ be a smooth function defined on $\overline{\mathcal{M}}$, and f be the restriction of $\bar{f}$ to $\mathcal{M}$. Then, the Riemannian gradient of f at point $X \in \mathrm{St}_S(m, l)$ is equal to the projection of* grad $\bar{f}$ *onto $T_X\mathcal{M}$, i.e.,*

$$\mathrm{grad}\, f(X) = \mathcal{P}_{T_X} \mathrm{grad}\, \bar{f}(X).$$

Remark 2.7. *It should be emphasized that the gradient operator* grad $\bar{f}(X)$ *differs fundamentally from the Euclidean gradient $\nabla \bar{f}(X)$, although the domain of* grad $\bar{f}(X)$ *is specifically the manifold* $\mathrm{St}_S(m, l)$*. This distinction stems from the particular geometric structure of the manifold for f, which possesses an intrinsically defined inner product that deviates from conventional Euclidean metrics.*

2.3. Seven-step framework of ZNN

In a survey published in 2024, Uhlig [19] elaborated the seven-step construction procedure for discrete ZNN models in detail. The specific steps are outlined as follows:

(1) Establish the objective function $E(t) = F(v_x(t), v_y(t), \dots) - G(A(t), B(t), \dots)$;
(2) Construct the error evolution equation $\dot{E}(t) = -\eta E(t)$;
(3) Combine the results of Steps (1) and (2) to solve for the time derivatives $\dot{v}_x(t), \dot{v}_y(t)$ of the time-varying projection vectors $v_x(t), v_y(t)$;
(4) Select a convergent finite difference formula;
(5) Substitute the outcome of Step (3) into the finite difference scheme from Step (4) to derive a new equality;
(6) Obtain the iterative sequences from Step (5):

$$v_x(t+1) = f\big(v_x(t), \dot{v}_x(t), \dots\big), \quad v_y(t+1) = g\big(v_y(t), \dot{v}_y(t), \dots\big);$$

(7) Repeat Steps (1)–(6) until the iterative sequences converge.

For Step (4), available convergent finite difference formulas include (but are not limited to) the following:

$$\begin{aligned}
\dot{y}_t &= \frac{3}{5\tau}y_{t+1} - \frac{3}{10\tau}y_t - \frac{1}{5\tau}y_{t-1} - \frac{1}{10\tau}y_{t-2},\\
\dot{y}_t &= \frac{5}{8\tau}y_{t+1} - \frac{3}{8\tau}y_t - \frac{1}{8\tau}y_{t-1} - \frac{1}{8\tau}y_{t-2},\\
\dot{y}_t &= \frac{1}{\tau}y_{t+1} - \frac{3}{2\tau}y_t + \frac{1}{\tau}y_{t-1} - \frac{1}{2\tau}y_{t-2},\\
\dot{y}_t &= \frac{1}{2\tau}y_{t+1} - \frac{1}{2\tau}y_{t-1},\\
\dot{y}_t &= \frac{1}{\tau}y_{t+1} - \frac{1}{\tau}y_t.
\end{aligned}$$

Two core components lie within the above seven-step solution framework:

(1) The error function decays exponentially to zero over time.
(2) This methodology adopts forward convergent finite difference equations instead of standard ordinary differential equation initial-value solvers. The first property guarantees the convergence and efficiency of the algorithm, while the second property enables discretization of the continuous-time dynamics and enhances practical applicability.

3. ZNN method for LDA

Prior to proposing the ZNN method for LDA, we first consider reformulating the original LDA optimization problem (2.2) in the following specific form:

$$\begin{aligned} \max_{Q} \quad & \text{trace}(Q^T S_b Q), \\ \text{s.t.} \quad & Q^T S_w Q = I. \end{aligned} \tag{3.1}$$

With respect to (3.1), we first aim to derive the expression of its Riemannian gradient, laying the groundwork for the proposal of the ZNN method.

Theorem 3.1. *Let $f(Q) = \text{trace}(Q^T S_b Q)$ be a mapping from $Q \in \text{St}_{S_w}(m, r)$ to $\mathbb{R}$. Then, the Riemannian gradient of f is given by*

$$\text{grad } f(Q) = (S_w^{-1} - QQ^T)(S_b + S_b^T)Q. \tag{3.2}$$

Proof. Suppose that $\bar{f}(Q) = \text{trace}(Q^T S_b Q)$ is defined on $\mathbb{R}^{m\times r}$. For $\forall\ \eta \in \text{St}_{S_w}(m, r)$, by Definition 2.4, we get

$$\langle \text{grad } \bar{f}(Q), \eta\rangle_Q = \text{D } \bar{f}(Q)[\eta] = \text{trace}(\eta^T (S_b + S_b{}^T)Q). \tag{3.3}$$

From Definition 2.3, we obtain

$$\langle \text{grad } \bar{f}(Q), \eta\rangle_Q = \text{ trace}(\eta^T S_w \text{grad}\bar{f}(Q)). \tag{3.4}$$

By (3.3) and (3.4), we have

$$\text{grad } \bar{f}(Q) = S_w^{-1}(S_b + S_b{}^T)Q. \tag{3.5}$$

According to Lemma 2.5, Lemma 2.6, and (3.5), we derive that

$$\begin{aligned} \text{grad } f(Q) &= \mathcal{P}_{T_Q}\text{grad}\bar{f}(Q) \\ &= S_w^{-1}(S_b + S_b{}^T)Q \\ &\quad - Q\text{sym}(Q^T S_w S_w^{-1}(S_b + S_b{}^T)Q) \\ &= (S_w^{-1} - QQ^T)(S_b + S_b^T)Q. \end{aligned}$$

□

The Riemannian gradient expression (3.2) is crucial for designing the ZNN method for (3.1). Specifically, building upon Theorem 3.1, the neural network model of problem (3.1) is formulated through the time-variant differential system

$$\frac{dQ(t)}{dt} = (S_w^{-1} - Q(t)Q(t)^T)(S_b + S_b^T)Q(t), \tag{3.6}$$

which is called the neural dynamical system of (3.1). Subsequently, temporal discretization via the forward Euler method yields the iterative update rule

$$Q^{(k+1)} = Q^{(k)} + \tau[(S_w^{-1} - Q^{(k)}(Q^{(k)})^T)(S_b + S_b^T)Q^{(k)}], \tag{3.7}$$

where τ is the sampling time, and $Q^{(k)} = Q(t = k\tau)$. The complete neural dynamical system of LDA is therefore algorithmically codified in Algorithm 1.

Algorithm 1 Riemannian gradient-based Zhang neural network method for LDA

Input: Given two matrices $S_b, S_w \in \mathbb{R}^{m\times m}$, initial values $Q^{(0)} \in \mathbb{R}^{m\times r}$, sampling time τ, the tolerance ε, and the maximum number of iterations $\mathrm{IT}_{\max}$.

1: Set $k = 0$.
2: Compute $R_1^{(k)} = (S_w^{-1} - Q^{(k)}(Q^{(k)})^T)(S_b + S_b^T)Q^{(k)}$.
3: **While** $\|R_1^{(k)}\|_2 > \varepsilon$, and $k < \mathrm{IT}_{\max}$ **do**
4: $\quad k = k + 1$.
5: $\quad Q^{(k)} = Q^{(k-1)} + \tau R_1^{(k-1)}$.
6: $\quad R_1^{(k)} = (S_w^{-1} - Q^{(k)}(Q^{(k)})^T)(S_b + S_b^T)Q^{(k)}$.
7: **end While**
8: **Output:** $Q = Q^{(k)}$.

4. Theoretical analysis

In the above section, we present the neural network method for LDA, i.e., Algorithm 1. In this section, we consider the theoretical analysis of this method.

Theorem 4.1. *If the initial value $Q(0) \in \mathrm{St}_{S_w}(m, r)$, then for all $t \geq 0$, the iterative sequence $\{Q(t)\} \in \mathrm{St}_{S_w}(m, r)$ generated by Algorithm 1.*

Proof. This theorem can be proved by verifying that for every initial point with $Q(t)^T S_w Q(t) = I$, the resulting dynamical trajectory satisfies the following equation:

$$\frac{d(Q(t)^T S_w Q(t))}{dt} = 0.$$

From the differential system (3.6), we get

$$\begin{aligned}
&\frac{d(Q(t)^T S_w Q(t))}{dt} \\
&= \left(\frac{dQ(t)}{dt}\right)^T S_w Q(t) + Q(t)^T S_w \left(\frac{dQ(t)}{dt}\right) \\
&= \left[(S_w^{-1} - Q(t)Q(t)^T)(S_b + S_b^T)Q(t)\right]^T S_w Q(t) \\
&\quad + Q(t)^T S_w \left[(S_w^{-1} - Q(t)Q(t)^T)(S_b + S_b^T)Q(t)\right] \\
&= Q(t)^T (S_b + S_b^T)Q(t) - Q(t)^T (S_b + S_b^T)Q(t)Q(t)^T S_w Q(t) \\
&\quad + Q(t)^T (S_b + S_b^T)Q(t) - Q(t)^T S_w Q(t)Q(t)^T (S_b + S_b^T)Q(t) \\
&= 0.
\end{aligned}$$

Then, we know that

$$\left(\frac{dQ(t)}{dt}\right)^T S_w Q(t) + Q(t)^T S_w \left(\frac{dQ(t)}{dt}\right) = 0,$$

which implies that $Q(t) \in \mathrm{St}_{S_w}(m, r)$ for all $t \geq 0$. □

Remark 4.2. *This theorem implies that if $Q(0) \in \mathrm{St}_{S_w}(m, r)$, then the sequence $\{Q(k)\}$ generated through the iterative procedure of Algorithm 1 is always in $\mathrm{St}_{S_w}(m, r)$.*

Next, we consider the stability of the Riemannian gradient-based ZNN method for LDA.

We first review some properties of the ordinary differential equation:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t)), \quad \mathbf{x}(t_0) \in \mathbb{R}^n, \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector. From [32], if $f : \mathbb{R}^n \to \mathbb{R}^n$ is continuous differentiable, then the solution of (4.1) exists for $t_0 < t$ and is unique for some given initial state $\mathbf{x}(t_0)$.

Definition 4.3. *[32] We call $\mathbf{x}_*$ an equilibrium point or fixed point of* (4.1) *if $f(\mathbf{x}_*) = 0$.*

Definition 4.4. *[32] For every neighborhood $\mathbb{U}$ of $\mathbf{x}_*$, if there exists a neighborhood $\mathbb{U}_1 \subset \mathbb{U}$ such that every solution $\mathbf{x}(t)$ with $\mathbf{x}(t_0) \in \mathbb{U}_1$ lies in $\mathbb{U}$ for every $t > t_0$, then $\mathbf{x}_*$ is stable.*

Definition 4.5. *[32] If $\mathbf{x}_*$ possesses the properties of Definition 4.4 and $\mathbb{U}_1$ can be chosen such that*

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}_*$$

for every $\mathbf{x}(t_0) \in \mathbb{U}_1$, then $\mathbf{x}_*$ is asymptotically stable.

In accordance with the Lyapunov direct method, we need to find a positive definite function known as the "Lyapunov function". This function decreases along the solution curves of the dynamical system. Then, the stability of equilibrium points can be ascertained by this function. The Lyapunov direct method [33, 34] can be given by the following lemma.

Lemma 4.6. *[33, 34] Let $\mathbf{x}_*$ be an equilibrium point of* (4.1) *and $F(\mathbf{x})$ be a continuous scalar function defined in a neighborhood $\mathbb{U}$ of $\mathbf{x}_*$, differentiable in $\mathbb{U} - \{\mathbf{x}_*\}$, and such that*

(1) $F(\mathbf{x}_*) = 0$ and $F(\mathbf{x}) > 0$ for $\mathbf{x} \neq \mathbf{x}_*$,

(2) if $\frac{dF(\mathbf{x})}{dt} \leq 0$ for $\mathbf{x} \in \mathbb{U} - \{\mathbf{x}_*\}$, then $\mathbf{x}_*$ is stable,

(3) if $\frac{dF(\mathbf{x})}{dt} < 0$ for $\mathbf{x} \in \mathbb{U} - \{\mathbf{x}_*\}$, then $\mathbf{x}_*$ is asymptotically stable.

Now, we consider the dynamical system of (3.6). Suppose that

$$F(Q) = \text{trace}(Q^T S_b Q),$$

and Q_* is a local maximizer of $F(Q)$. Obviously, $F(Q)$ is continuous and differentiable on $\text{St}_{S_w}(m, r)$. Then, there exist a constant $\varepsilon > 0$ and a neighborhood

$$\mathbb{U}(Q_*, \varepsilon) = \{Q \in \text{St}_{S_w}(m, r) : \|Q - Q_*\| \leq \varepsilon\}$$

such that

$$F(Q) \leq F(Q_*)$$

for all $Q \in \mathbb{U}(Q_*, \varepsilon)$.

The following theorem demonstrates the local stability of the proposed ZNN method.

Theorem 4.7. *If the initial value $Q(0) \in \mathbb{U}(Q_*, \varepsilon)$, where Q_* is a local maximizer of $F(Q)$ in $\text{St}_{S_w}(m, r)$, then the solution of the neural dynamical system* (3.6) *is local stable in the sense of Lyapunov stability theory at Q_*.*

Proof. Without loss of generality, we first assume that $m > r$. In order to prove that $\text{trace}(Q(t)^T S_b Q(t)) \leq \text{trace}(Q_*{}^T S_b Q_*)$ holds for any $t \geq 0$ and $Q(t) \in \mathbb{U}(Q_*, \varepsilon)$, we define the function:

$$\psi(t) = -\text{trace}(Q(t)^T S_b Q(t)) + \text{trace}(Q_*{}^T S_b Q_*).$$

From Theorem 4.1, for all $t \geq 0$, it follows that $Q(t) \in \text{St}_{S_w}(m, r)$, and thus $\psi(t) \geq 0$.

Without loss of generality, assume that the within-class scatter matrix S_w is non-singular. By Eq (2.1), S_w is a symmetric positive definite matrix and S_b is a symmetric matrix. Therefore, the Cholesky decomposition can be performed:

$$S_w = LL^T.$$

Then, the constraint is transformed into

$$Q(t)^T S_w Q(t) = Q(t)^T LL^T Q(t) = I_r.$$

Let $D(t) = L^T Q(t)$, hence

$$D(t)^T D(t) = I_r, \quad Q(t) = L^{-T} D(t).$$

Letting $D(t)_\perp$ be the orthogonal complements of $D(t)$, we have

$$D(t)D(t)^T + D(t)_\perp D(t)_\perp^T = I_m.$$

By direct calculations, we prove that

$$\begin{aligned}
\frac{d\psi(t)}{dt} &= \frac{d[-\text{trace}(Q(t)^T S_b Q(t))]}{dt} = -2\text{trace}\left(\frac{dQ(t)^T}{dt} S_b Q(t)\right) \\
&= -2\text{trace}\left(\left((S_w^{-1} - Q(t)Q(t)^T)(S_b + S_b{}^T)Q(t)\right)^T S_b Q(t)\right) \\
&= -4\text{trace}\left(Q(t)^T S_b{}^T \left(S_w^{-1} - Q(t)Q(t)^T\right) S_b Q(t)\right) \\
&= -4\text{trace}\left(Q(t)^T S_b{}^T \left(L^{-T}L^{-1} - L^{-T}D(t)D(t)^T L^{-1}\right) S_b Q(t)\right) \\
&= -4\text{trace}\left(Q(t)^T S_b{}^T L^{-T} \left(I_m - D(t)D(t)^T\right) L^{-1} S_b Q(t)\right) \\
&= -4\text{trace}\left[Q(t)^T S_b{}^T L^{-T}\right. \\
&\quad \left. \cdot \left(I_m - D(t)D(t)^T - D(t)_\perp D(t)_\perp^T + D(t)_\perp D(t)_\perp^T\right) L^{-1} S_b Q(t)\right] \\
&= -4\text{trace}\left(Q(t)^T S_b{}^T L^{-T} D(t)_\perp D(t)_\perp^T L^{-1} S_b Q(t)\right) \\
&= -4\left\|Q(t)^T S_b{}^T L^{-T} D(t)_\perp\right\|_F^2 \leq 0,
\end{aligned}$$

which implies that the solution of the neural dynamical system (3.6) is local stable. □

5. Numerical experiments

In this section, we demonstrate the effectiveness of Algorithm 1 through several specific numerical experiments. All numerical computations were implemented in MATLAB R2023a on a hardware platform configured with a 3.90 GHz AMD Ryzen 5 5600 CPU and 32 GB RAM.

It is crucial to note that all computational tests systematically compare Algorithm 1 with a counterpart named WH2. The WH2 algorithm, first proposed in [22], is specifically designed for

singular value decomposition computations. The WH2 algorithm mentioned in this article is an extension of this idea to solving LDA.

The derivation process of the WH2 algorithm mainly consists of two steps. First, for the optimization problem (3.1), the Lagrangian function is defined by

$$L(Q, \Lambda) = \text{trace}(Q^T S_b Q) - \frac{1}{2}\langle \Lambda, Q^T S_w Q - I \rangle,$$

where $\Lambda \in \mathbb{R}^{r\times r}$ is a Lagrangian multiplier matrix. It is easy to show that Λ is Hermitian and the global optimal solution satisfies the following equation:

$$\begin{cases} \left.\dfrac{\partial L(Q,\Lambda)}{\partial Q}\right|_{Q=Q_*} = (S_b + S_b^T)Q_* - S_w Q_* \Lambda_* = 0, \\ Q_*^T S_w Q_* = I_r. \end{cases}$$

Through simple derivation, we obtain

$$\Lambda_* = Q_*^T (S_b + S_b^T) Q_*.$$

Second, similar to (3.6), we establish a neural dynamical system,

$$\frac{dQ(t)}{dt} = (I - S_w Q(t) Q(t)^T)(S_b + S_b^T) Q(t). \tag{5.1}$$

Similar to Algorithm 1, we can also provide pseudocode for the WH2 algorithm, but we will not specifically provide it here.

We use the norm of gradients to measure the error of algorithms. Specially, we define the following metric of the Riemannian gradient of Algorithm 1:

$$\varepsilon(Q(t)) = \left\| (S_w^{-1} - Q(t)Q(t)^T)(S_b + S_b^T)Q(t) \right\|_2.$$

Similarly, we define the metric of the gradient of WH2 algorithm as follows:

$$\varepsilon(Q(t)) = \left\| (I - S_w Q(t)Q(t)^T)(S_b + S_b^T)Q(t) \right\|_2.$$

Furthermore, we quantify the degree to which the constraint condition $Q(t)^T S_w Q(t) = I_r$, which characterizes the generalized orthogonality of $Q(t)$, is satisfied, as follows:

$$n(Q(t)) = \|Q(t)^T S_w Q(t) - I\|_2.$$

The Riemann gradient norm $\varepsilon(Q)$, as an optimality measure, is used to evaluate the optimal solution of the iteration. The generalized orthogonality constraint error $n(Q)$ acts as a geometric validity metric, measuring whether the iterated projection matrix strictly satisfies the intrinsic manifold constraint of LDA on the generalized Stiefel manifold.

For the two algorithms, we use two metrics to measure their efficiency: the iteration steps and CPU time, denoted as "IT" and "CPU", respectively. To ensure more accurate results, in all subsequent numerical experiments, we repeat each test 20 times and record the averaged values. The stopping criterion for both algorithms is either $\varepsilon(Q(t))$ <1e-7 or IT = $\text{IT}_{\max}$ = 80000.

5.1. Synthetic data

In this subsection, we test Algorithm 1 and WH2 using synthetic data.

Example 5.1. *Let $m = 90, d = 120$ and $r = 20$. First, we consider generating two matrices X and Y as follows:*

$$X = A + 0.1 \cdot (C + Z)Y,$$

where $A \in \mathbb{R}^{m\times d}$ is a Gaussian random matrix, the matrix $Y \in \mathbb{R}^{m\times d}$ has independent entries that take value ± 1 with equal probability, $C \in \mathbb{R}^{m\times m}$ is a matrix whose entries are all 1, and $Z \in \mathbb{R}^{m\times m}$ has independent entries from the uniform distribution $[0, 1]$. On this basis, we construct matrices S_b and S_w as follows:

$$S_b = XX^T, \quad S_w = YY^T.$$

The above four matrices and the initial states $Q(0)$ can be generated using MATLAB's built-in functions in the following ways:

```
A=randn(m,d),  B=sign(rand(m,d)-0.5*ones(m,d)),  C=ones(m,m),  Z=rand(m,m),

Q(0)=B'\orth(rand(d,r)).
```

The sampling time was set to six different values, respectively, and the experimental results are recorded in Table 1.

Table 1. Numerical results for Examples 5.1.

Method	τ	IT	CPU	$\varepsilon(Q)$	$n(Q)$
Algorithm 1	1e-2	-	-	-	-
	1e-3	80000	16.5346	99.6768	0.9668
	1e-4	80000	15.8175	3.1462e-4	1.1140e-11
	1e-5	80000	15.4384	0.1262	6.7048e-06
	1e-6	80000	17.1063	2.7769	0.3701
	1e-7	80000	16.6273	23.3507	0.5647
WH2	1e-2	-	-	-	-
	1e-3	-	-	-	-
	1e-4	-	-	-	-
	1e-5	-	-	-	-
	1e-6	80000	18.1362	0.9068	5.5763e-3
	1e-7	80000	19.8295	32.0945	0.2898

The symbol "-" in Table 1 represents algorithm divergence, which means that the error increases as the iteration progresses. From Table 1, it can be observed that when the sampling time τ=1e-6 or smaller, 80,000 iterations are insufficient to make the error function $\varepsilon(Q(t))$ converge to the specified accuracy. When τ=1e-5 and τ=1e-4, the WH2 algorithm remains divergent, but $\varepsilon(Q(t))$ in Algorithm 1 decrease to 3.1462e-4, while $n(Q(t))$ is reduced to 1e-11. Furthermore, $\varepsilon(Q(t))$ will gradually decrease

as the iteration progresses until it reaches the stop criterion. At this point, the manifestation of generalized orthogonality can be observed. When τ=1e-3, the accuracy of error functions Algorithm 1 deteriorates. For τ=1e-2, both algorithms diverge.

In other words, when the sampling time is excessively small, the algorithm converges slowly, preventing $\varepsilon(Q(t))$ from reaching the convergence accuracy of 1e-7 within 80,000 iterations. Conversely, an excessively large sampling time causes $\varepsilon(Q(t))$ to diverge. This naturally give rise to two questions: First, can we find an appropriate sampling time that enables $\varepsilon(Q(t))$ for both algorithms to converge to 1e-7? Second, if $\varepsilon(Q(t))$ of both algorithms can converge to 1e-7, what is the relationship between the sampling time and the convergence time? And can we identify the sampling time corresponding to the minimal convergence time? If feasible, we refer to this sampling time as the "fastest-converging sampling time".

In fact, in Example 5.1, the fastest-converging sampling time for Algorithm 1 is 6e-4, whereas the WH2 algorithm fails to converge $\varepsilon(Q(t))$ to 1e-7 within 80,000 iterations at any sampling time. For the WH2 algorithm, 3e-6 yields the fastest relative convergence rate, with experimental results presented in Table 2. When operating at their respective fastest-converging sampling time, Algorithm 1 outperforms WH2 algorithm in both convergence accuracy and convergence time.

Table 2. Numerical results for Examples 5.1.

Method	τ	IT	CPU	$\varepsilon(Q)$	$n(Q)$
Algorithm 1	6e-4	36220	7.8519	9.9961e-08	1.6279e-14
WH2	3e-6	80000	18.2031	8.9613e-03	2.6076e-05

5.2. Real data

In this subsection, we test the effectiveness of Algorithm 1 and the WH2 algorithm using some real datasets.

5.2.1. Image reconstruction

We selected the ORL dataset and used them to evaluate the performance of Algorithm 1 and WH2 algorithm in image reconstruction* [35]. This dataset contains 400 face images of 40 persons, with varying expressions and poses. The pixels of all the images are 92×112. Figure 1 shows some of the original images from the ORL dataset.

*https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Figure 1. A portion of the images from the ORL dataset.

Example 5.2. *We extract matrices from the 400 original grayscale images in the ORL dataset. The between-class scatter matrix S_b and the within-class scatter matrix S_w are generated by the following formula:*

$$\begin{aligned} S_b &= \sum_{i=1}^{c} n_i(\bar{P}_i - \bar{P})^T(\bar{P}_i - \bar{P}), \\ S_w &= \sum_{i=1}^{c}\sum_{j=1}^{n_i}(P_j^{(i)} - \bar{P}_i)^T(P_j^{(i)} - \bar{P}_i), \end{aligned} \tag{5.2}$$

where

$$\begin{aligned} \bar{P} &= \frac{1}{N}\sum_{i=1}^{c}\sum_{j=1}^{n_i} P_j^{(i)}, \\ \bar{P}_i &= \frac{1}{n_i}\sum_{j=1}^{n_i} P_j^{(i)}, \quad i = 1, 2, \ldots, c. \end{aligned} \tag{5.3}$$

Here, $\bar{P}$ and $\bar{P}_i$ represent the average value of the overall training samples and the average value of the training samples of the i-th class, respectively. Moreover, n_i denotes the number of training samples of the i-th class, and it satisfies $n_i + \cdots + n_c = N$. $P_j^{(i)}$ represents the j-th training sample of the i-th class in the training set.

For the convenience of presenting the experimental results, we use image "1.jpg" as an example to demonstrate the image reconstruction of both algorithms. Denoting the matrix of grayscale image "1.jpg" as P_1, we obtain its Fisher feature matrix $\widetilde{P}_1$ as follows:

$$\widetilde{P}_1 = P_1 Q.$$

Then, the reconstructed image of $\widehat{P}_1$ can be expressed as

$$\widehat{P}_1 = \widetilde{P}_1 Q^{\dagger}. \tag{5.4}$$

Since S_w is a symmetric positive definite matrix, we perform a Cholesky decomposition $S_w = LL^T$. From the constraint condition $Q(t)^T S_w Q(t) = I$, it follows that $Q(t)^T LL^T Q(t) = I$. Then, the initial states $Q(0)$ can be generated by

```
Q(0)=L'\orth(rand(m,r)).
```
(5.5)

We know that $N = 400, c = 40, n_1 = n_2 = \cdots = n_{40} = 10, S_b \in \mathbb{R}^{92\times92}$, and $S_w \in \mathbb{R}^{92\times92}$. Then, we set $\tau = 0.09$ and $\tau =$ 1.4e-5 for Algorithm 1 and the WH2 algorithm, respectively. The results are presented in Figures 2 and 3, respectively.

Figure 2. The effect of image reconstruction for Algorithm 1. Five images, from left to right, are denoted as "original image", "$r = 23$", "$r = 46$", "$r = 69$", "$r = 92$".

Figure 3. The effect of image reconstruction for WH2 algorithm. Five images, from left to right, are denoted as "original image", "$r = 23$", "$r = 46$", "$r = 69$", "$r = 92$".

From Figures 2 and 3, it can be observed that as the number of axes r in the Fisher projection matrix Q increases, the reconstructed image $\widehat{P}$ becomes increasingly similar to the original image. When r approaches the image dimension, the quality of the reconstructed image is almost the same as the original image. The results indicate that both Algorithm 1 and the WH2 algorithm are applicable to image reconstruction, and when all the Fisher projection axes are utilized, the original image can be completely reconstructed.

Figure 4 compares the convergence curves of Algorithm 1 and the WH2 algorithm for Example 5.2 with rank $r = 69$. The left panel plots the Riemannian gradient norm $\varepsilon(Q(t))$ measuring optimality, while the right panel presents the generalized orthogonality constraint error $n(Q(t))$ for manifold geometric feasibility. We observe that Algorithm 1 converges much faster on both metrics. Its Riemannian gradient norm rapidly decreases below 10^{-4}, and the orthogonality constraint error falls to around 10^{-7}. However, the WH2 algorithm converges slowly and suffers from severe stagnation: Its constraint error remains above 10^{-2} over all iterations. This demonstrates that the proposed Riemannian ZNN can obtain solutions that simultaneously satisfy optimality conditions and the generalized Stiefel manifold constraint, whereas WH2 fails to precisely maintain the geometric constraint in the LDA optimization task.

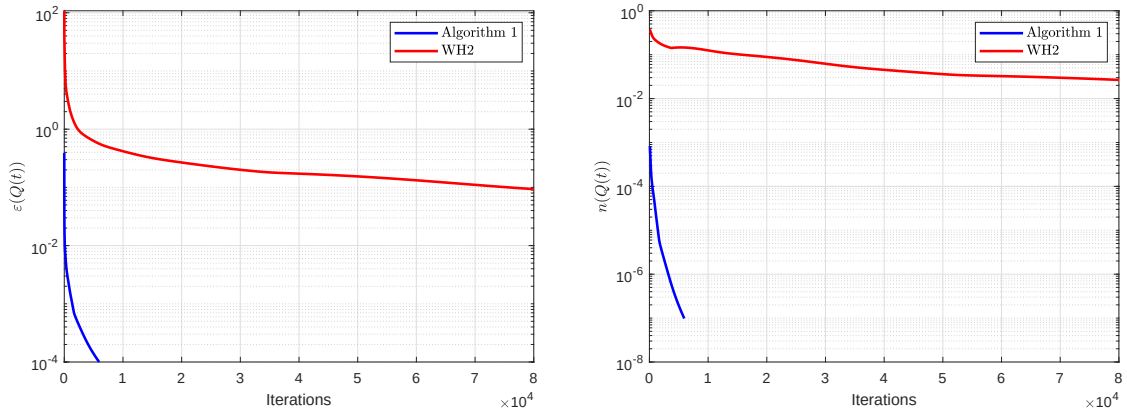

Figure 4. Comparison of convergence curves for two algorithms under Example 5.2 with rank $r = 69$.

5.2.2. Image reconstruction rate

According to Eq (5.4), we can naturally calculate the reconstruction rate of the image "1.jpg" as follow

$$\text{Ratio} = 1 - \frac{\left\|\widehat{P}_1 - P_1\right\|_2}{\|P_1\|_2}.$$

Based on the reconstruction rate, we can measure the performance of Algorithm 1 and the WH2 algorithm in image reconstruction.

In order to compare the image reconstruction rates of Algorithm 1 and the WH2 algorithm, we first set some relevant parameters, as shown in Table 3. Figure 5 shows the changes in the reconstruction rate of the image "1.jpg" as r varies, for both algorithms, when their fastest-converging sampling time is maintained. The settings for sampling time and stopping criteria are as shown in Table 3. Here, "CPU" in Table 3 indicates that when $r = 1$ to $r = 92$, the total convergence time of Algorithm 1 is 358 seconds.

Table 3. Experimental setup for image reconstruction rate.

Method		τ	stopping criteria	CPU
Algorithm 1	Riman-1	0.09	1e-4	358.2931
	Riman-2	0.09	1e-7	1505.8344
WH2	WH2	1.4e-5	1e-7	3826.7852

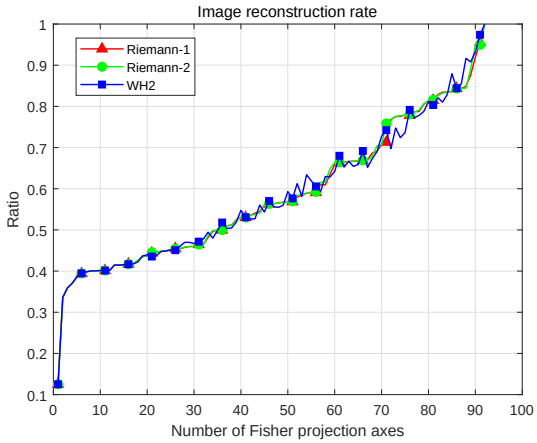

Figure 5. The image reconstruction rates of the two algorithms.

From Figure 5, the curves "Riemann-1" and "Riemann-2" are almost overlapping. This implies that as long as the error function $\varepsilon(Q(t))$ can converge to 10^{-4}, the reconstruction rate of the image "1.jpg" can reach its maximum. The convergence time of "Riemann-1" is 358 seconds, which is significantly faster than that of "Riemann-2" (1505 seconds). For the WH2 algorithm, the general trend of the curve is consistent with that of Algorithm 1, but local instability has emerged in the image reconstruction rate. This, to some extent, demonstrates the advantages of Algorithm 1 over the WH2 algorithm.

Next, we examine the influence of the stopping criteria on the reconstruction rate. We set the stopping criteria as $\varepsilon(Q(t)) < 10^{-2}$, $\varepsilon(Q(t)) < 10^{-3}$ and $\varepsilon(Q(t)) < 10^{-4}$ respectively, and the corresponding reconstruction rate curves are denoted as "Riemann-0.01", "Riemann-0.001" and "Riemann-0.0001". From Figure 6, it can be seen that when $\varepsilon(Q(t)) > 10^{-4}$, the reconstruction rate is fluctuating and unstable. Only when $\varepsilon(Q(t)) < 10^{-4}$, the image reconstruction rate will gradually increase as r increases. This indicates that setting the stop criterion at 10^{-4} is the optimal choice.

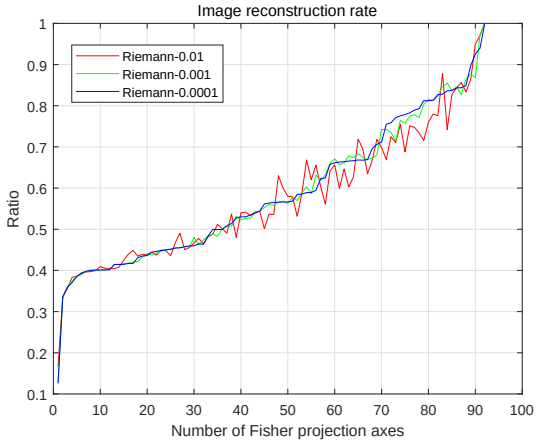

Figure 6. The influence of different stopping criteria on the image reconstruction rate.

5.2.3. The influence of the sampling time

In what follows, we tested the fastest-converging sampling time for Algorithm 1 using Example 5.2. The values of sampling time τ were selected as 50 different numbers: 0.01, 0.02, 0.03,. . ., 0.50, the stopping criterion was $\varepsilon(Q(t)) < 10^{-4}$, and $r = 60$. The result is presented in Figure 7.

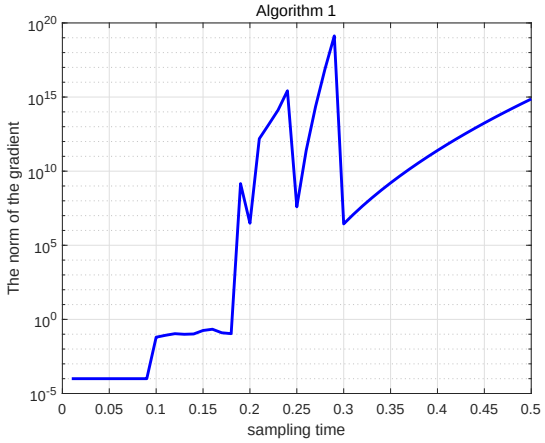

Figure 7. The influence of different sampling times on Algorithm 1.

From Figure 7, when sampling time $\tau > 0.18$, Algorithm 1 diverges. When $0.1 \leq \tau \leq 0.18$, $\varepsilon(Q(t))$ converges to 1e-2. Only when $\tau \leq 0.09$ can Algorithm 1 converge to 1e-4. From Table 4, clearly, when $\tau = 0.09$, the CPU is smaller than others. Therefore, we infer that in Examples 5.2, $\tau = 0.09$ is the fastest-converging sampling time for Algorithm 1.

Table 4. The influence of different sampling times on Algorithm 1.

τ	iterations	CPU
0.01	98655	64.0132
0.02	49325	31.2099
0.03	32882	19.7653
0.04	24660	15.1359
0.05	19727	12.3229
0.06	16438	10.0987
0.07	14089	9.2622
0.08	12327	7.6921
0.09	10957	6.6668

In other words, in the image reconstruction rate experiment, the sampling time of 0.09 and the stopping criterion $\varepsilon(Q(t)) < 10^{-4}$ represent the optimal configuration for Algorithm 1. Under this configuration, as shown in Table 3, when r ranges from 1 to 92, the fastest time required for Algorithm 1 to complete the experiment is 358 seconds.

6. Conclusions

In this paper, we propose a ZNN method based on Riemannian gradients for solving linear discriminant analysis. It should be pointed out that the existing theoretical derivation assumes that the within-class scatter matrix S_w is positive definite and free of noise, while real-world datasets generally contain outliers, label noise, and singular cases under small-sample-size scenarios. In the next step, we can combine perturbation analysis to derive the bias bound of the Riemannian gradient when S_b and S_w are disturbed by noise, and construct a robust Riemannian ZNN model with a perturbation compensation term. Aiming at the singular problem of S_w under small samples, we integrate the idea of regularized LDA (RLDA), embed an adaptive regularization term into the manifold optimization objective, and complete the corresponding stability proof. Furthermore, we will explore the introduction of appropriate nonlinear activation functions into the proposed neural dynamic system. Reasonably designed activation functions are expected to strengthen the nonlinear mapping capability of the ZNN model, suppress oscillations during the iterative process, and further improve the convergence performance of the algorithm on complex data.

Author contributions

Yu-Hang Wang: Writing – review & editing, writing – original draft, validation, methodology, investigation, formal analysis; Zhuo-Cheng Xie: Writing – review & editing, supervision, methodology, investigation, formal analysis, conceptualization; Meng-Zhen Fu: Supervision, conceptualization, methodology, formal analysis; Huan Ren: Supervision, methodology, funding acquisition, formal analysis, conceptualization. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Funding declaration

The work is supported by the National Natural Science Foundation of China under Grant No. 12401496, the Natural Science Foundation of Jiangxi Province under Grant Nos. 20242BAB20006 and 20244BCE52256, the Doctoral Research Initiation Fund of Jiangxi Science and Technology Normal University under Grant No. 2023BSQD25, and the Jiangxi Provincial Postgraduate Innovation Special Fund under Grant No. YC2025-S197.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

1. R. A. Fisher, The use of multiple measurements in taxonomic problems, *Ann. Eugen.*, **7** (1936), 179–188. https://doi.org/10.1111/j.1469-1809.1936.tb02137.x
2. R.O. Duda, P. E. Hart, *Pattern classification and scene analysis*, Wiley, 1973.
3. R.O. Duda, P. E. Hart, D. G. Stork, *Pattern classification*, 2 Eds., Wiley, 2000.
4. R. Lotlikar, R. Kothari, Fractional-step dimensionality reduction, *IEEE Trans. Pattern Anal. Mach. Intell.*, **22** (2000), 623–627. https://doi.org/10.1109/34.862200
5. P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, Eigenfaces vs. fisherfaces: Recognition using class specific linear projection, *IEEE Trans. Pattern Anal. Mach. Intell.*, **19** (1997), 711–720. https://doi.org/10.1109/34.598228
6. K. Liu, Y. Q. Cheng, J. Y. Yang, A generalized optimal set of discriminant vectors, *Pattern Recogn.*, **25** (1992), 731–739. https://doi.org/10.1016/0031-3203(92)90136-7
7. K. Liu, Y. Q. Cheng, J. Y. Yang, X. Liu, An efficient algorithm for Foley-Sammon optimal set of discriminant vectors by algebraic method, *Int. J. Pattern Recogn. Artif. Intell.*, **6** (1992), 817–829. https://doi.org/10.1142/S0218001492000412
8. H. Yu, J. Yang, A direct lda algorithm for high-dimensional data-with application to face recognition, *Pattern Recogn.*, **34** (2001), 2067–2070. https://doi.org/10.1016/S0031-3203(00)00162-X
9. M. Loog, R. P. W. Duin, R. Haeb-Umbach, Multiclass linear dimension reduction by weighted pairwise fisher criteria, *IEEE Trans. Pattern Anal. Mach. Intell.*, **23** (2001), 762–766. https://doi.org/10.1109/34.935849
10. J. Lu, K. N. Plataniotis, A. N. Venetsanopoulos, Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition, *Pattern Recognit. Lett.*, **26** (2005), 181–191. https://doi.org/10.1016/j.patrec.2004.09.014
11. D. H. Foley, J. W. Sammon, An optimal set of discriminant vectors, *IEEE Trans. Comput.*, **24** (1975), 281–289. https://doi.org/10.1109/T-C.1975.224208
12. Z. Jin, J. Y. Yang, Z. S. Hu, Z. Lou, Face recognition based on the uncorrelated discriminant transformation, *Pattern Recogn.*, **34** (2001), 1405–1416. https://doi.org/10.1016/S0031-3203(00)00084-4
13. W. K. Ching, D. Chu, L. Z. Liao, X. Wang, Regularized orthogonal linear discriminant analysis, *Pattern Recogn.*, **45** (2012), 2719–2732. https://doi.org/10.1016/j.patcog.2012.01.007
14. M. Meilă, H. Zhang, Manifold learning: What, how, and why, *Annu. Rev. Stat. Appl.*, **11** (2024) 393–417. https://doi.org/10.1146/annurev-statistics-040522-115238

15. J. Qi, W. Shuai, Y. Lv, M. Yang, S. Jin, HyperDiffuseNet: A deep hyperbolic manifold learning method for dimensionality reduction in spatial transcriptomics, *J. Comput. Biol.*, **32** (2025) 1101–1120. https://doi.org/10.1177/15578666251377097

16. Y. Zhou, S. Sun, Manifold partition discriminant analysis, *IEEE Trans. Cybern.*, **47** (2017), 830–840. https://doi.org/10.1109/TCYB.2016.2529299

17. Z. Yao, Z. Wang, D. Wang, J. Wu, L. Chen, An ensemble CNN-LSTM and GRU adaptive weighting model based improved sparrow search algorithm for predicting runoff using historical meteorological and runoff data as input, *J. Hydrol.*, **625** (2023) 129977. https://doi.org/10.1016/j.jhydrol.2023.129977

18. Y. Zhang, C. Yi, *Zhang neural networks and neural-dynamic method*, New York: Noval Science Publishers, 2011.

19. F. Uhlig, Zhang neural networks: An introduction to predictive computations for discretized time-varying matrix problems, *Numer. Math.*, **156** (2024), 691–739. https://doi.org/10.1007/s00211-023-01393-5

20. X. Wang, M. Che, Y. Wei, Recurrent neural network for computation of generalized eigenvalue problem with real diagonalizable matrix pair and its applications, *Neurocomputing*, **216** (2016), 230–241. https://doi.org/10.1016/j.neucom.2016.07.042

21. X. Wang, J. Shan, Y. Wei, Neural networks for total least squares solution of the time-varying linear systems, *Comput. Appl. Math.*, **44** (2025), 106. https://doi.org/10.1007/s40314-024-03070-1

22. A. Weingessel, K. Hornik, SVD algorithms: APEX-like versus subspace methods, *Neural Process. Lett.*, **5** (1997), 177–184. https://doi.org/10.1023/A:1009642710601

23. B. Huang, W. Li, Neural network models for the quaternion singular value decompositions, *Appl. Math. Model.*, **129** (2024), 780–805. https://doi.org/10.1016/j.apm.2024.02.019

24. Y. Gan, D. Zhou, Iterative methods based on low-rank matrix for solving the Yang Baxter-like matrix equation, *Comput. Appl. Math.*, **43** (2024), 241. https://doi.org/10.1007/s40314-024-02771-x

25. H. Zhang, L. Wan, Zeroing neural network methods for solving the Yang-Baxter-like matrix equation, *Neurocomputing*, **383** (2020), 409–418. https://doi.org/10.1016/j.neucom.2019.11.101

26. J. Zhao, Y. Zhang, A novel zeroing neural network with integral upper-bound function for solving time-varying Sylvester equation, *Comput. Appl. Math.*, **44** (2025), 374. https://doi.org/10.1007/s40314-025-03340-6

27. Y. Wei, P. Stanimirović, M. Petković, *Numerical and symbolic computations of generalized inverses*, New Jersey: World Scientific, 2018.

28. Y. Zhang, D. Guo, *Zhang functions and various models*, Berlin, Heidelberg: Springer, 2015. https://doi.org/10.1007/978-3-662-47334-4

29. Z. C. Xie, M. Wang, Y. H. Wang, H. Ren, Riemannian-based neural network method for solving canonical correlation analysis, *Comput. Appl. Math.*, **44** (2025), 234. https://doi.org/10.1007/s40314-025-03197-9

30. P. A. Absil, R. Mahony, R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, 2008.

31. X. Wang, K. Deng, Z. Peng, C. Yan, New vector transport operators extending a Riemannian CG algorithm to generalized Stiefel manifold with low-rank applications, *J. Comput. Appl. Math.*, **451** (2024), 116024. https://doi.org/10.1016/j.cam.2024.116024

32. J. Zabczyk, *Mathematical control theory: An introduction*, Cham: Birkhäuser, 2020. https://doi.org/10.1007/978-3-030-44778-6

33. M. Hirsch, S. Smale, *Differential equations, dynamical systems and linear algebra*, San Diego: Academic Press, 1974.

34. J. P. La Salle, *The stability of dynamical systems*, Philadelphia: SIAM, 1976.

35. F. Samaria, A. Harter, Parameterisation of a stochastic model for human face identification, In: *Proceedings of 1994 IEEE workshop on applications of computer vision*, 1994. https://doi.org/10.1109/ACV.1994.341300

AIMS Press