# Neural Fisher Discriminant Analysis: Optimal Neural Network Embeddings in Polynomial Time

**Burak Bartan** [1] **Mert Pilanci** [1]

## Abstract

Fisher's Linear Discriminant Analysis (FLDA) is a statistical analysis method that linearly embeds data points to a lower dimensional space to maximize a discrimination criterion such that the variance between classes is maximized while the variance within classes is minimized. We introduce a natural extension of FLDA that employs neural networks, called Neural Fisher Discriminant Analysis (NFDA). This method finds the optimal two-layer neural network that embeds data points to optimize the same discrimination criterion. We use tools from convex optimization to transform the optimal neural network embedding problem into a convex problem. The resulting problem is easy to interpret and solve to global optimality. We evaluate the method's performance on synthetic and real datasets.

## 1. Introduction

In this work, we develop a novel natural extension of Fisher's Linear Discriminant Analysis (FLDA) using neural network embeddings. The method of FLDA is used to find a linear combination of features that separates two or more classes (Fisher, 1936). An advantage of FLDA is that it does not make any distribution assumptions on the data. This is in contrast to the method of Linear Discriminant Analysis (LDA), which makes multivariate Gaussian assumptions for the class conditional probabilities. FLDA aims to find a projection that minimizes the sample variance within classes and maximizes the sample variance across classes in the projected space. The resulting projection has various useful interpretations for dimension reduction and classification problems.

In this work, we develop a non-linear neural network based variant of FLDA, which we term Neural Fisher Discriminant Analysis (NFDA). Our method aims to combine the representation power of neural networks with the optimal separation criterion of the FLDA method. We provide algorithms for finding the optimal weights for the two-layer neural network transformation that maximizes the ratio of between-class variance to within-class variance for binary and multiclass problems. We show by proving matching lower and upper bounds that our algorithms find the global optimum of this objective.

The proposed NFDA method involves different algorithms depending on the activation function considered. We consider ReLU activation $g(t) = \max(0, t)$ and degree-two polynomial activations $g(t) = at^2 + bt + c$ in this work. Furthermore, we present the NFDA method for both binary class and multiclass problems. We start by introducing the binary case, since the multiclass case requires a more sophisticated analysis due to the vector valued second layer weights. This results in the multiclass NFDA method generating $K - 1$ times more neurons for a multiclass problem with $K$ classes compared to the binary class NFDA method. Furthermore, the resulting algorithms correspond to the standard FLDA problem on a modified version of the input. Specifically, we apply *feature lifting* to the input. The goal of feature lifting is to transform the input to a higher dimension where non-linear relations between features are included as features. The type of feature lifting depends on the activation function.

### 1.1. Related Work

Degree-two polynomial activations have a significant computational advantage over other activation functions. In particular, it is possible to reformulate the training problem with degree-two activations into a tractable convex program that is solvable in fully-polynomial time as shown in (Bartan & Pilanci, 2021a;b). Moreover, a compact parameterization of the neural network parameter space considerably simplifies the neural network training problem. In this work, we explore a similar parameterization approach. The NFDA method for polynomial activation returns a more compact neural network compared to the NFDA method for ReLU. Furthermore, polynomial activation functions have found

[1] Department of Electrical Engineering, Stanford University, CA, USA. Correspondence to: Burak Bartan <bbartan@stanford.edu>, Mert Pilanci <pilanci@stanford.edu>.

*Proceedings of the $39^{th}$ International Conference on Machine Learning*, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

practical use in recent work including neural network inference on encrypted data (Gilad-Bachrach et al., 2016) and are shown to perform comparable to other activation functions (Allen-Zhu & Li, 2020) such as ReLU. Our derivation of the ReLU activation NFDA leverages a recent convex analytical framework developed for ReLU networks (Pilanci & Ergen, 2020; Wang et al., 2021; Ergen & Pilanci, 2020; 2021a;c;d;b; 2019). These results were extended to ReLU networks with vector outputs (Sahiner et al., 2020), batch norm (Ergen et al., 2021), generative adversarial networks (Sahiner et al., 2021a) and autoencoders (Sahiner et al., 2021b; Gupta et al., 2021).

Neural network based Fisher discriminant analysis methods have been proposed in the literature in works including (Stuhlsatz et al., 2012; Wu et al., 2017; Díaz-Vico & Dorronsoro, 2020; Dorfer et al., 2016). The proposed techniques in these works mostly require training neural networks using backpropagation. In contrast, in this work we prove that optimal two-layer neural network transformations could be obtained by feature lifting and solving a linear system, followed by eigenvalue decomposition if it is a multiclass problem. Hence, our proposed methodology does not depend on any heuristics and guarantees globally optimal solutions. Moreover, there exist other non-linear versions of Fisher discriminant analysis in the literature such as kernel discriminant analysis (Mika et al., 1999), (Centeno & Lawrence, 2006).

### 1.2. Notation

For neural network weights, we use $w_j^{(1)}$ for first layer weights and $w_j^{(2)}$ for second layer weights for the $j$'th neuron in the hidden layer. Also, we use $\theta := \{w_j^{(1)}, w_j^{(2)}\}_{j=1}^m$ to represent all of the parameters of the network. The notation $\mathbb{1}[\cdot] : \mathbb{R} \to \{0, 1\}$ denotes the indicator function and it outputs 1 if its argument is true and 0 otherwise. We will use $[K]$ to denote the set of integers $\{1, \dots, K\}$.

The notation $D = \text{Diag}(v)$ is used to refer to the diagonal matrix with diagonal entries equal to the entries of its input, $D_{jj} = v_j$. Moreover, $\text{vec}(\cdot)$ denotes the vectorized version of its argument. We will use $e_k$ for the $k$'th unit vector of the appropriate dimension. The $i$'th sample of the dataset is $x_i$ while the notation $\bar{x}_i$ corresponds to the feature lifted version of the $i$'th sample. The exact feature lifting depends on the activation function and is defined explicitly in the text. Similarly, $X$ is the input matrix whose rows are different samples and $\bar{X}$ is the feature lifted dataset.

## 2. Preliminaries

### 2.1. Fisher's Linear Discriminant Analysis

Suppose $x_1, \dots, x_n \in \mathbb{R}^d$ represent observations from two classes. Let the class labels be such that $y_i \in \{0, 1\}$. We will use $X \in \mathbb{R}^{n \times d}$ to denote the data matrix where the $i$'th row of $X$ is the $i$'th data sample $x_i$. Let us define the sample mean vectors and covariance matrices for each class $k = 0, 1$ as follows:

$$\begin{aligned}\mu_k &:= \mathbb{E}_X[x \,|\, x \sim \text{ class } k] \\ &= \frac{1}{n_k} \sum_{y_i = k} x_i \,, \qquad (1) \\ \Sigma_k &:= \mathbb{E}_X[(x - \mu_k)(x - \mu_k)^T \,|\, x \sim \text{ class } k] \\ &= \frac{1}{n_k - 1} \sum_{y_i = k} (x_i - \mu_k)(x_i - \mu_k)^T \,, \qquad (2)\end{aligned}$$

where we used the notation $\mathbb{E}_X$ for the empirical mean with respect to the dataset $X$ and $\mathbb{E}_X[\cdot \,|\, x \sim \text{ class } k]$ is the empirical class conditional expectation when the label of the sample $x$ is equal to class $k$. Also, $n_k$ is the number of samples in the $k$'th class.

In FLDA, we consider finding a linear feature map $w \in \mathbb{R}^d$ such that the ratio of the variance between classes to the variance within the classes is maximized in the projection space $w^T x$. Note that the class conditional mean $\beta_k$ and variance $\sigma_k^2$ of the mapped features $z = w^T x$ are given by:

$$\begin{aligned}\beta_k &:= w^T \mu_k, \qquad (3) \\ \sigma_k^2 &:= w^T \Sigma_k w, \qquad (4)\end{aligned}$$

for $k = 0, 1$. Then, the ratio of the sample variances can be written as $(\beta_0 - \beta_1)^2 / (\sigma_0^2 + \sigma_1^2)$. The goal of FLDA is to find the transformation that maximizes this ratio:

$$w^* = \arg\max_w \frac{(\beta_0 - \beta_1)^2}{\sigma_0^2 + \sigma_1^2} \,. \qquad (5)$$

This can be written as a ratio of two quadratics:

$$w^* = \arg\max_w \frac{w^T (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w}{w^T (\Sigma_0 + \Sigma_1) w} \,. \qquad (6)$$

An optimal solution to this problem is given by $w^* = c(\Sigma_0 + \Sigma_1)^{-1}(\mu_0 - \mu_1)$ where $c \in \mathbb{R}$ is an arbitrary non-zero constant.

### 2.2. Multiclass Fisher's LDA

Consider a dataset with $K$ classes and let the class labels be such that $y_i \in \{0, \dots, K-1\}$. In multiclass FLDA, the goal is to find a linear transformation $z = W^T x$ where $W \in \mathbb{R}^{d \times (K-1)}$ such that the within-class scatter is minimized while the between-class scatter is maximized. Before we give the exact form of this objective, we define the within-class $S_W$ and between-class scatter matrices $S_B$ as follows:

$$S_W := \sum_{k=0}^{K-1} S_k, \text{ where } S_k := \sum_{y_i=k} (x_i - \mu_k)(x_i - \mu_k)^T \tag{7}$$

$$S_B := \sum_{k=0}^{K-1} n_k(\mu_k - \mu)(\mu_k - \mu)^T \tag{8}$$

where $\mu$ is the mean of all the samples, $\mu := \frac{1}{n}\sum_{i=1}^n x_i$. The scatter matrices in the transformed domain are equal to $W^T S_W W$ and $W^T S_B W$, respectively.

Multiclass FLDA problem is typically formulated as a maximization of the ratio of the determinants of the scatter matrices in the transformed domain:

$$W^* = \arg\max_W \frac{\det(W^T S_B W)}{\det(W^T S_W W)}\,, \tag{9}$$

where $\det(\cdot)$ is the matrix determinant. The optimal transformation $W^*$ is given by the eigenvectors of $S_W^{-1}S_B$ corresponding to the largest $K-1$ eigenvalues. More specifically, the $j$'th column of $W^*$ is equal to the $j$'th eigenvector of $S_W^{-1}S_B$ when the eigenvectors are sorted in descending order of their corresponding eigenvalues.

# 3. Neural Fisher Discriminant Analysis for Binary Class Problems

Recall that in Fisher's LDA, the goal is to find the optimal linear transformation that maximizes the ratio of variances. It is easy to see that a more expressive mapping than linear can potentially improve the performance due to increased representational power. In particular, we will consider two-layer neural networks

$$f(x) := \sum_{j=1}^m g(x^T w_j^{(1)}) w_j^{(2)} \tag{10}$$

where $w_j^{(1)} \in \mathbb{R}^d$ and $w_j^{(2)} \in \mathbb{R}$, $j = 1, \ldots, m$ denote the first and second layer weights. $g : \mathbb{R} \to \mathbb{R}$ is the activation function. Let $\theta := \{w_j^{(1)}, w_j^{(2)}\}_{j=1}^m$ denote all of the parameters of the network. Let us define the sample mean and variance in the neural network domain:

$$\beta_k(\theta) := \mathbb{E}_X[f(x)\,|\,x \sim \text{ class } k]\,, \tag{11}$$

$$\sigma_k^2(\theta) := \mathbb{E}_X[(f(x) - \beta_k(\theta))^2\,|\,x \sim \text{ class } k]\,. \tag{12}$$

Finding the neural network transformation $f$ (parameterized by $\theta$) that maximizes the ratio of the between-class variance to within-class variance can be posed as the following optimization problem:

$$\theta^* = \arg\max_\theta \frac{(\beta_0(\theta) - \beta_1(\theta))^2}{\sigma_0^2(\theta) + \sigma_1^2(\theta)}\,. \tag{13}$$

In the remainder of this section, we present algorithms for solving this optimization problem optimally. In the sequel, we will refer to this problem as the NFDA problem. To the best of our knowledge, global optimization of this objective to find optimal neural networks has not been considered in the literature. We note that the solution of this problem depends on the type of activation function $g$. We first focus on the solution when $g$ is the ReLU activation and then consider the polynomial activation.

## 3.1. ReLU Activation

Let the activation function $g$ be the ReLU activation $g(t) = (t)_+ = \max(0, t)$. We will show that the optimal solution to the NFDA problem in (13) can be obtained via applying a specific feature lifting to the input and solving the FLDA on the lifted features. The details of the method are given in Algorithm 1. Theorem 3.1 states the main result for NFDA for binary class problems for ReLU networks.

The feature lifting step of Algorithm 1 requires multiplying the input by the diagonal matrices $D_j, j = 1, \ldots, P$ defined as

$$D_j := \text{Diag}(\mathbb{1}[X\bar{h}_j \geq 0]), \tag{14}$$

where $\mathbb{1}[X\bar{h}_j \geq 0]$ is an $n$-dimensional vector of 0's and 1's. Intuitively, multiplying the input matrix $X$ by these 0-1 valued diagonal matrices $D_j$ corresponds to masking certain rows of the data, which parallels the action of ReLU activation on its input. A full explanation can be found in the proof of Theorem 3.1.

Note that the number of unique matrices $D_j$ depends on the input matrix $X$. It is easy to see that a low-rank matrix $X$ will have a low number of unique $D_j$'s. A simple way to obtain the matrices $D_j$ is to randomly sample i.i.d. $\bar{h}_j$ vectors from the standard normal distribution and evaluate $\text{Diag}(\mathbb{1}[X\bar{h}_j \geq 0])$. We denote the number of distinct matrices $D_j$ by $P$. It is known that the maximal number of distinct matrices $D_j = \text{Diag}(\mathbb{1}[X\bar{h}_j \geq 0])$ is upper bounded by

$$P \leq 2r\Big(\frac{e(n-1)}{r}\Big)^r\,. \tag{15}$$

Here, $r := \text{rank}(X) \leq d$, and the above number is polynomial in $n$ for any fixed matrix rank $r$, or dimension $d$. In (Pilanci & Ergen, 2020), the enumeration of $D_j$ arises in formulating convex programs for training two-layer ReLU networks. Theorem 3.1 assumes that all $P$ distinct $D_j$ matrices are enumerated, which can be done in time $O(n^r)$ (Edelsbrunner et al., 1986). We note that as $\text{rank}(X)$ grows, this will become infeasible. Hence, in the numerical simulations, we rely on an approximation by considering $P' < P$ distinct $D_j$ matrices. This approximation has been shown to still capture the representational power of ReLU sufficiently well in recent works including (Pilanci & Ergen,

**Algorithm 1** Binary NFDA for ReLU activation

```
    **Input:** Dataset $X \in \mathbb{R}^{n\times d}, y \in \mathbb{R}^n$
    Generate $\bar{h}_j, j = 1, \dots, P$
    Compute $\bar{X} = \begin{bmatrix} D_1 X & \dots & D_P X \end{bmatrix} \in \mathbb{R}^{n \times dP}$
    Form mean vectors and covariance matrices for the lifted input samples $\bar{x}_i \in \mathbb{R}^{dP}$:
        $\bar{\mu}_k = \frac{1}{n_k} \sum_{y_i = k} \bar{x}_i$        (16)
        $\bar{\Sigma}_k = \frac{1}{n_k - 1} \sum_{y_i = k} (\bar{x}_i - \bar{\mu}_k)(\bar{x}_i - \bar{\mu}_k)^T$        (17)
    Solve the system $\bar{w}^* = (\bar{\Sigma}_0 + \bar{\Sigma}_1)^{-1}(\bar{\mu}_0 - \bar{\mu}_1)$
    Solve the linear program (LP) in (21)
    **Output:** Optimal weights:
        $(w_j^{(1)}, w_j^{(2)}) = \begin{cases} (u_j^*, 1) & 1 \le j \le P \\ (v_{j-P}^*, -1) & P < j \le 2P \end{cases}$        (18)
```

2020; Sahiner et al., 2020). As will be shown in the numerical results section, picking a low $P'$ leads to good performance in our setting as well.

**Theorem 3.1** (NFDA for binary class and ReLU activation)**.** *Algorithm 1 solves the NFDA problem in* (13) *for ReLU activation* $g(t) = \max(0, t)$ *optimally when the number of neurons satisfies* $m \ge m^*$. *The threshold* $m^*$ *is equal to*

$$m^* = \sum_{j=1}^{P} (\mathbb{1}[u_j^* \neq 0] + \mathbb{1}[v_j^* \neq 0])\,, \tag{19}$$

*where* $u_j^*$ *and* $v_j^*$ *are the solutions to the LP in* (21).

We now give the proof sketch for Theorem 3.1, which consists of two main components; finding a upper bounding tractable optimization problem via relaxations and decomposing the solution of the upper bounding problem to prove a matching lower bound. The full proof is provided in the appendix.

**Upper bound:** First, we note that the ReLU activation can be equivalently written as a dot product times the indicator function $(x^T w_j^{(1)})_+ = (x^T w_j^{(1)})\,\mathbb{1}[x^T w_j^{(1)} \ge 0]$. Next, we relax the NFDA problem by introducing the variables $h_j$ and replacing the indicator term with $\mathbb{1}[x^T h_j \ge 0]$. Let us denote the optimal value of the original NFDA problem by $p_1^*$ and the relaxed problem by $p_2^*$. It follows that we have $p_2^* \ge p_1^*$. We relax the problem again by considering the enumeration of all possible unique vectors $\mathbb{1}[X\bar{h}_j \ge 0]$. The solution of the relaxed problem (with optimal value $p_3^*$) gives an upper bound for the optimal value of the original problem, $p_3^* \ge p_2^* \ge p_1^*$. The resulting transformation after the relaxations is equivalent to a linear transformation on the lifted input. More specifically, we obtain the transformation $f_{r2}(x)$ given as

$$f_{r2}(x) = \underbrace{\begin{bmatrix} {w_1^{(1)}}^T & \dots & {w_P^{(1)}}^T \end{bmatrix}}_{\bar{w}^T} \underbrace{\begin{bmatrix} x\,\mathbb{1}[x^T \bar{h}_1 \ge 0] \\ \vdots \\ x\,\mathbb{1}[x^T \bar{h}_P \ge 0] \end{bmatrix}}_{\bar{x}} \tag{20}$$

where $\bar{x} \in \mathbb{R}^{dP}$ is the lifted input and $\bar{w} \in \mathbb{R}^{dP}$ is the vector of weights. Hence, we can solve the relaxed optimization problem optimally by solving a linear system. We show that this linear system is the same as the problem that arises in FLDA except that it is in the lifted input space. The solution of the FLDA problem for the lifted data $\bar{X}$ gives us an upper bound on the optimal value of the original problem.

**Lower bound:** The main strategy behind establishing the lower bound is decomposing the solution $\bar{w}^*$ obtained from solving the upper bounding FLDA problem such that we arrive at a feasible set of weights for a two-layer ReLU network. This cone decomposition method has first been proposed and analyzed in recent work (Mishkin et al., 2022).

We formulate an auxiliary optimization problem that decomposes the blocks ${w_j^{(1)}}^*$ of $\bar{w}^*$ into $u_j \in \mathbb{R}^d$ and $v_j \in \mathbb{R}^d$ for $j = 1, \dots, P$ such that the signs of $(x_i^T u_j)$ and $(x_i^T v_j)$ match the signs of $(x_i^T \bar{h}_j)$. More explicitly, this corresponds to solving the following linear program

$$\begin{aligned} \text{find} \quad & u_j, v_j \\ \text{s.t.} \quad & u_j - v_j = {w_j^{(1)}}^*, \ \forall j \in [P] \\ & (2D_j - I) X u_j \ge 0, \ \forall j \in [P] \\ & (2D_j - I) X v_j \ge 0, \ \forall j \in [P]. \end{aligned} \tag{21}$$

The solution of the LP provides neural network weights as given in (18). Note that the objective evaluates to $p_3^*$ for these weights. Since any two-layer ReLU network weights are at most as good as the optimal weights, the weights in (18) are in fact a lower bounding solution, that is, $p_3^* \le p_1^*$. Hence, we obtain $p_3^* = p_1^*$ and this concludes the proof that Algorithm 1 optimally solves the NFDA problem in (13).

## 3.2. Polynomial Activation

Suppose now that the hidden layer activation is the second degree polynomial activation function $g(t) = at^2 + bt + c$ where $a, b, c \in \mathbb{R}$ are tunable coefficients. For polynomial activation, the high-level strategy to solve the NFDA problem follows that of ReLU activation. The type of feature lifting and the recovery method for the neural network weights are the main differences between the NFDA methods for polynomial activation and ReLU activation. In particular,

**Algorithm 2** Binary NFDA for polynomial activation

```
Input: Dataset $X \in \mathbb{R}^{n\times d}, y \in \mathbb{R}^n$
Compute lifting $\bar{X} = \begin{bmatrix} \bar{x}_1^T \\ \vdots \\ \bar{x}_n^T \end{bmatrix} \in \mathbb{R}^{n\times(d^2+d+1)}$, where
$\bar{x}_i = \begin{bmatrix} a\,\mathrm{vec}(x_i x_i^T) \\ b x_i \\ c \end{bmatrix} \in \mathbb{R}^{d^2+d+1}$
Form mean vectors and covariance matrices for the lifted input as in (16)
Solve the system $\bar{w}^* = (\bar{\Sigma}_0 + \bar{\Sigma}_1)^{-1}(\bar{\mu}_0 - \bar{\mu}_1)$
Construct $Z^*$ from $\bar{w}^*$ as in (22)
Compute the eigendecomposition $Z^* = \sum_j u_j u_j^T \alpha_j$ where $u_j$'s are eigenvectors and $\alpha_j$'s are eigenvalues.
Output: Optimal weights:
    $(w_j^{(1)}, w_j^{(2)}) = (c_j/d_j, \alpha_j d_j^2), \quad j = 1, \ldots, m^*$   (23)
where $u_j = \begin{bmatrix} c_j \\ d_j \end{bmatrix} \in \mathbb{R}^{d+1}$ and $c_j \in \mathbb{R}^d$, $d_j \in \mathbb{R}$
```

the feature lifting for polynomial activation is a quadratic function. Furthermore, the optimal weights are constructed via eigenvalue decomposition.

The steps of the proposed method are listed in Algorithm 2. After obtaining the linear system solution $\bar{w}^*$, we recover neural network weights by computing the eigenvalue decomposition of $Z^*$ which is constructed by reshaping $\bar{w}^*$:

$$Z^* := \begin{bmatrix} Z_1^* & Z_2^* \\ Z_2^{*T} & Z_4^* \end{bmatrix} = \begin{bmatrix} \mathrm{vec}^{-1}(\bar{w}^*_{1:d^2}) & \bar{w}^*_{d^2+1:d^2+d} \\ \bar{w}^{*T}_{d^2+1:d^2+d} & \bar{w}^*_{d^2+d+1} \end{bmatrix}, \tag{22}$$

where the notation $w_{i:j}$ is used to point to the entries of $w$ from the $i$'th index to the $j$'th. Here, the operator $\mathrm{vec}^{-1}$ stands for the inverse vectorization operation which reshapes the first $d^2$ entries of $\bar{w}^*$ into a $d \times d$ matrix. The blocks of the matrix $Z^* \in \mathbb{R}^{(d+1)\times(d+1)}$ have the following dimensions: $Z_1^* \in \mathbb{R}^{d\times d}, Z_2^* \in \mathbb{R}^d, Z_4^* \in \mathbb{R}$.

Theorem 3.2 states the main result for the NFDA method for polynomial activation.

**Theorem 3.2** (NFDA for binary class and polynomial activation)**.** *Algorithm 2 solves the problem (13) for polynomial activation $g(t) = at^2 + bt + c$ optimally when the number of neurons satisfies $m \geq m^*$. The threshold $m^*$ is equal to*

$$m^* = \mathrm{rank}(Z^*)\,, \tag{24}$$

*where $Z^*$ is as defined in (22).*

The proof of Theorem 3.2 is provided in the appendix. The proof relies on the compact parameterization of polynomial activation networks in terms of the eigendecomposition. More specifically, we show that the neural network transformation can be written as

$$\begin{aligned} f(x) &= \sum_{j=1}^{m} g(x^T w_j^{(1)}) w_j^{(2)} \\ &= \Bigg\langle \underbrace{\begin{bmatrix} a\,\mathrm{vec}(xx^T) \\ bx \\ c \end{bmatrix}}_{\bar{x}}, \underbrace{\begin{bmatrix} \mathrm{vec}\left(\sum_{j=1}^m w_j^{(1)} w_j^{(1)^T} w_j^{(2)}\right) \\ \sum_{j=1}^m w_j^{(1)} w_j^{(2)} \\ \sum_{j=1}^m w_j^{(2)} \end{bmatrix}}_{\bar{w}} \Bigg\rangle, \end{aligned} \tag{25}$$

where $\langle \cdot, \cdot \rangle$ indicates dot product. We note that this is unlike the ReLU network where we relaxed the problem multiple times before we obtain a linear transformation on the lifted input. In this case, we show that we can directly obtain a linear transformation on the quadratically lifted input.

## 4. Neural Fisher Discriminant Analysis for Multiclass Problems

Note that for multiclass problems, second layer weights are vector valued. More specifically, we have $f(x) := \sum_{j=1}^m g(x^T w_j^{(1)}) w_j^{(2)}$ where $w_j^{(1)} \in \mathbb{R}^d$ and $w_j^{(2)} \in \mathbb{R}^{K-1}$ denote the first and second layer weights. We will use $\theta$ to represent the weights of the neural network as before. Sample mean and covariance matrix for the transformed data points in class $k$ are as follows

$$\mu_k(\theta) = \mathbb{E}_X[f(x) \,|\, x \sim \text{class } k]\,, \tag{26}$$

$$\begin{aligned} &\Sigma_k(\theta) = \\ &= \mathbb{E}_X[(f(x) - \mu_k(\theta))(f(x) - \mu_k(\theta))^T \,|\, x \sim \text{class } k]\,. \end{aligned} \tag{27}$$

Next, we define the within-class $S_W(\theta)$ and between-class scatter matrices $S_B(\theta)$ as follows

$$S_W(\theta) := \sum_{k=0}^{K-1} S_k(\theta), \quad \text{where}$$

$$S_k(\theta) := \sum_{y_i = k} (f(x_i) - \mu_k(\theta))(f(x_i) - \mu_k(\theta))^T \tag{28}$$

$$S_B(\theta) := \sum_{k=0}^{K-1} n_k \big(\mu_k(\theta) - \mu(\theta)\big)\big(\mu_k(\theta) - \mu(\theta)\big)^T \tag{29}$$

where $\mu(\theta) = \frac{1}{n}\sum_{i=1}^n f(x_i)$. Now, we are ready to present the optimization problem for finding the optimal neural network transformation that maximizes the ratio of the between-class scatter to the within-class scatter:

$$\theta^* = \arg\max_{\theta} \frac{\det(S_B(\theta))}{\det(S_W(\theta))}\,. \tag{30}$$

**Algorithm 3** Multiclass NFDA for ReLU activation

```
**Input:** Dataset $X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n$
Generate $\bar{h}_j, j = 1, \dots, P$
Compute lifting $\bar{X} = \begin{bmatrix} D_1 X & \dots & D_P X \end{bmatrix} \in \mathbb{R}^{n \times dP}$
Compute the mean vectors and covariance matrices for the lifted input as in (16) and $\bar{\mu} = \frac{1}{n} \sum_{i=1}^n \bar{x}_i$
Form the scatter matrices
    $\bar{S}_W = \sum_{k=0}^{K-1} \bar{\Sigma}_k,$
    $\bar{S}_B = \sum_{k=0}^{K-1} n_k (\bar{\mu}_k - \bar{\mu})(\bar{\mu}_k - \bar{\mu})^T \quad (31)$
Solve the system $\bar{S}_W^{-1} \bar{S}_B$ and find $\bar{W}^*$ by computing the eigenvectors corresponding to largest $K-1$ eigenvalues
Solve the linear program in (33)
**Output:** Optimal weights:
    $(u^*_{j,k}, e_k)$ and $(v^*_{j,k}, -e_k), \forall j \in [P], \forall k \in [K-1] \quad (32)$
```

This is the main optimization problem that we focus on in this section. We will show how we approach and solve this problem for the ReLU activation first and then polynomial activation. The techniques that we have used for the binary class problems in proving matching upper and lower bounds are mostly applicable in the multiclass case. The main difference from the binary class problem is that the multiclass formulation requires an additional step of optimization problem relaxation.

### 4.1. ReLU Activation

We now give the algorithm for solving the multiclass NFDA problem in (30) for ReLU activation and the corresponding main result. The steps are given in Algorithm 3. Note that the feature lifting is the same as the binary class NFDA algorithm. Differently from the binary class case, we compute the eigendecomposition of $\bar{S}_W^{-1} S_B$ for solving the FLDA for the lifted input. Moreover, note that the binary NFDA method can generate at most $2P$ neurons while multiclass NFDA could generate up to $2P(K-1)$ neurons.

The LP for recovering the neural network weights is as follows:

$$\begin{aligned} &\text{find} \;\; u_{j,k}, v_{j,k} \\ &\text{s.t.} \;\; u_{j,k} - v_{j,k} = w^{(1)^*}_{j,k}, \;\forall j \in [P], \,\forall k \in [K-1] \\ &\quad (2D_j - I) X u_{j,k} \geq 0, \;\forall j \in [P], \forall k \in [K-1] \\ &\quad (2D_j - I) X v_{j,k} \geq 0, \;\forall j \in [P], \forall k \in [K-1], \end{aligned} \tag{33}$$

where the variables are $u_{j,k}, v_{j,k} \in \mathbb{R}^d$, $j = 1, \dots, P$, $k = 1, \dots, K-1$. Here, $w^{(1)^*}_{j,k}$ denote the blocks of $\bar{W}^*$. Denote the solution to this LP by $u^*_{j,k}, v^*_{j,k}$. The optimal weights are then as given in (32). The main result is stated in Theorem 4.1. The full proof is given in the appendix.

**Theorem 4.1** (Multiclass NFDA for ReLU activation)**.** *Algorithm 3 solves the multiclass NFDA problem (30) for ReLU activation $g(t) = \max(0, t)$ optimally when the number of neurons satisfies $m \geq m^*$. The threshold $m^*$ is equal to*

$$m^* = \sum_{j=1}^{P} \sum_{k=1}^{K-1} \left( \mathbb{1}[u^*_{j,k} \neq 0] + \mathbb{1}[v^*_{j,k} \neq 0] \right), \tag{34}$$

*where $u^*_{j,k}$ and $v^*_{j,k}$ are the solutions to the LP in (33).*

### 4.2. Polynomial Activation

Unlike the binary NFDA for polynomial activation, we need to relax the problem to arrive at a linear transformation on the lifted input. After the relaxation, we obtain the following transformation

$$\begin{aligned} &f_r(x) = \\ &\sum_{k=1}^{K-1} \left\langle \underbrace{\begin{bmatrix} a \operatorname{vec}(xx^T) \\ bx \\ c \end{bmatrix}}_{\bar{x}}, \underbrace{\begin{bmatrix} \operatorname{vec}\left( \sum_{j=1}^m w^{(1)}_{j,k} w^{(1)^T}_{j,k} w^{(2)}_{j,k} \right) \\ \sum_{j=1}^m w^{(1)}_{j,k} w^{(2)}_{j,k} \\ \sum_{j=1}^m w^{(2)}_{j,k} \end{bmatrix}}_{\bar{w}_k} \right\rangle e_k \\ &= \bar{W}^T \bar{x} \end{aligned} \tag{35}$$

where the lifted input is $\bar{x} \in \mathbb{R}^{d^2+d+1}$ and the weight matrix is $\bar{W} \in \mathbb{R}^{(d^2+d+1) \times (K-1)}$. The columns of $\bar{W}$ are denoted $\bar{w}_k \in \mathbb{R}^{d^2+d+1}$.

The resulting method is listed in Algorithm 4. Theorem 4.2 gives the main result for multiclass NFDA for polynomial activation.

**Theorem 4.2** (Multiclass NFDA for polynomial activation)**.** *Algorithm 4 solves the multiclass NFDA problem (30) for polynomial activation $g(t) = at^2 + bt + c$ optimally when the number of neurons satisfies $m \geq m^*$. The threshold $m^*$ is equal to*

$$m^* = \sum_{k=1}^{K-1} \operatorname{rank}(Z^*_k), \tag{37}$$

*where $Z^*_k$ is as defined in Algorithm 4.*

The proof of Theorem 4.2 is given in the appendix. The proof idea combines the techniques used in proving Theorem 3.2 for binary class polynomial activation NFDA and Theorem 4.1 for multiclass ReLU activation NFDA.

**Algorithm 4** Multiclass NFDA for polynomial activation

```
**Input:** Dataset $X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n$
Compute lifting $\bar{X} = \begin{bmatrix} \bar{x}_1^T \\ \vdots \\ \bar{x}_n^T \end{bmatrix} \in \mathbb{R}^{n \times (d^2+d+1)}$, where
$\bar{x}_i = \begin{bmatrix} a\,\mathrm{vec}(x_i x_i^T) \\ b x_i \\ c \end{bmatrix} \in \mathbb{R}^{d^2+d+1}$
Compute mean vectors and covariance matrices for the lifted input as in (16)
Form the scatter matrices as in (31)
Solve the system $\bar{S}_W^{-1}\bar{S}_B$ and find $\bar{W}^*$ by computing the eigenvectors corresponding to largest $K-1$ eigenvalues
Construct $Z_k^*$ from columns $\bar{w}_k^*$ of $\bar{W}^*$ for every $k = 1, \ldots, K-1$ using (22)
For every $k$, compute the eigendecomposition $Z_k^* = \sum_j u_{j,k} u_{j,k}^T \alpha_{j,k}$ where $u_{j,k}$'s are eigenvectors and $\alpha_{j,k}$'s are eigenvalues.
**Output:** Optimal weights:
$$(c_{j,k}/d_{j,k}, \alpha_{j,k} d_{j,k}^2 e_k), \ \forall k \in [K-1], \forall j \in [\mathrm{rank}(Z_k^*)] \quad (36)$$
where $u_{j,k} = \begin{bmatrix} c_{j,k} \\ d_{j,k} \end{bmatrix} \in \mathbb{R}^{d+1}$, and $c_{j,k} \in \mathbb{R}^d$, $d_{j,k} \in \mathbb{R}$
```

## 5. Numerical Results

In this section, we provide numerical simulation results on various binary and multiclass datasets. We have implemented the NFDA algorithms in Python. In the numerical implementation of the NFDA methods, it is important to note that the linear system solution step of the algorithms can be prone to numerical stability issues. A common solution to this issue is to introduce regularization. For instance, for the multiclass NFDA method, this corresponds to solving the linear system $(\bar{S}_W + \beta I)^{-1}\bar{S}_B$, where $\beta > 0$ is the regularization coefficient. Finally, we use CVXPY (Diamond & Boyd, 2016) to numerically solve the LP that appears in the NFDA methods for ReLU activation.

### 5.1. Two Spirals Dataset

We have tested the performance of the binary NFDA method on the two spirals dataset (Chalup & Wiklendt, 2007) shown in Figure 1. This dataset has two classes where the blue crosses correspond to class 0 and orange circles correspond to class 1. The version of the dataset that we have used in our simulations has 120 samples. The samples are $d = 2$ dimensional. We have run the FLDA and NFDA methods on this dataset. These methods reduce the dimension of the samples to 1 since this is a binary class problem. Then, we plot the histograms of the tranformed samples as shown in Figure 2. For the NFDA method, we have used $P' = 100$ unique $D_j$ matrices. This figure demonstrates that the transformation due to NFDA moves the two classes further apart compared to FLDA. This is as expected since NFDA transformation is more general and has higher representation power compared to FLDA.

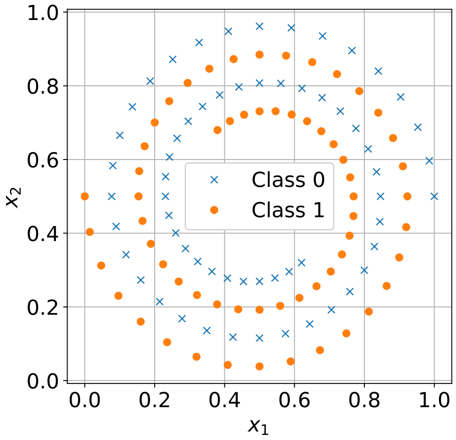
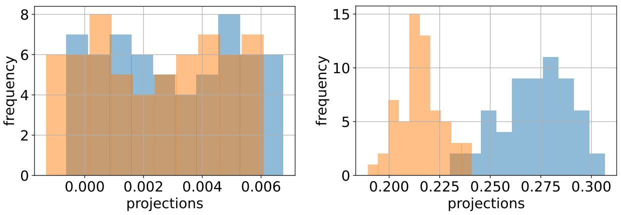


*Figure 1.* Two spirals dataset.



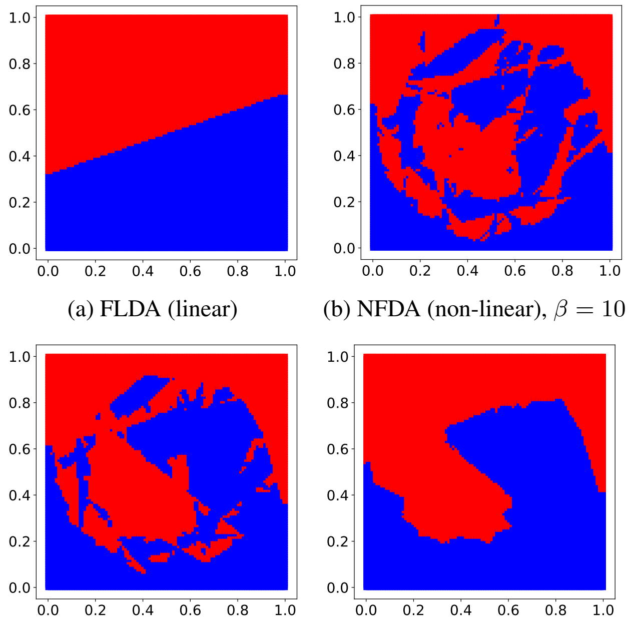
(a) FLDA (linear)

(b) NFDA (non-linear)

*Figure 2.* Histograms of the transformed samples of the two spirals dataset for FLDA and NFDA. Blue and orange colors correspond to classes 0 and 1, respectively.

To incorporate a bias term into the model, we have concatenated a constant term to each data sample, i.e. $[x_i^T; 1]$. This corresponds to modifying the model such that the input to the neurons in the hidden layer is $x^T w_j^{(1)} + b_j$.

We have plotted the resulting decision boundaries in Figure 3. The classification of a test sample after dimension reduction is determined by proximity to the mean of the two classes in the transformed domain. The FLDA decision boundary is linear since FLDA is a linear model. We have plotted the decision boundaries for NFDA for different regularization coefficient choices $\beta$ in plots b, c, d of Figure 3. As we increase $\beta$ from plot b to d, the decision boundary becomes less noisy as expected.



(a) FLDA (linear)

(b) NFDA (non-linear), $\beta = 10$

(c) NFDA (non-linear), $\beta = 10$

(d) NFDA (non-linear), $\beta = 100$

*Figure 3.* Visualization of decision boundaries for the two spirals dataset. $\beta$ is the regularization coefficient.

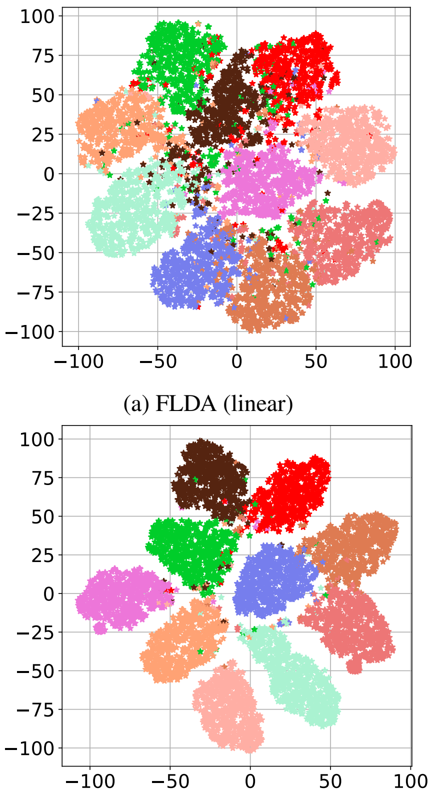
### 5.2. MNIST Dataset

We have tested the performance of the multiclass NFDA on the MNIST handwritten digit recognition dataset (LeCun et al., 2010). Each data sample is a $28 \times 28$ dimensional gray-scale image that shows a handwritten digit from 0 to 9.

We have used all of the $n = 60000$ training samples of the MNIST dataset in this simulation. Both FLDA and NFDA methods map data points into a 9-dimensional space since there are 10 classes. We have generated t-SNE (van der Maaten & Hinton, 2008) plots for the 9-dimensional outputs of these methods to visualize in 2 dimensions the separation between classes in the projected space. The resulting t-SNE plots are shown in Figure 4. For the NFDA method, we have used the ReLU activation with $P' = 20$. These figures show that the separation among classes is better for the proposed NFDA method compared to FLDA.

### 5.3. Face Images Dataset

The Yale Face dataset contains a total of 165 grayscale images of people's faces. The dataset includes 11 images per subject where each of these images corresponds to different configurations such as different lighting, glasses or no glasses, sleepy face expression, etc. For this simulation, we have resized the images to $18 \times 24$ pixels and we have selected 5 subjects. The class labels indicate different subjects. Consequently, we work with a 5-class dataset which contains 11 images for each class.



(a) FLDA (linear)

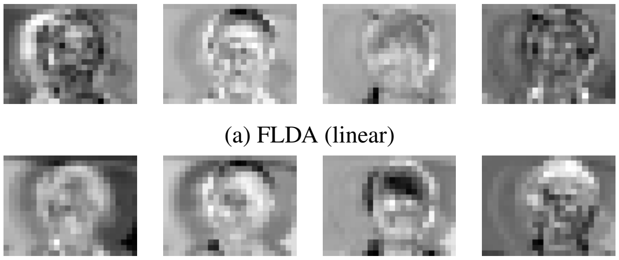
(b) NFDA (non-linear)

*Figure 4.* t-SNE plots for MNIST. Samples of the same class are shown using the same color.

We have plotted the projection directions resulting from the FLDA method as images in Figure 5-a. For the NFDA method, we have clustered the neurons into 4 groups and the centroids of the clusters are shown as images in Figure 5-b. We have used $P' = 15$ for the NFDA method with ReLU activation. The projection directions when viewed as images show the discriminative features of the images. The images for NFDA look more diverse compared FLDA, demonstrating the discriminative power of the method. The NFDA method is able to find more discriminative features due to the neural network structure.

## 6. Conclusion

We have introduced the NFDA problem for two-layer neural networks. The goal is to find the optimal two-layer neural network weights that maximize the between-class scatter while minimizing the within-class scatter analogous to the Fisher's linear discriminant analysis problem. We have then developed a methodology for solving this problem for ReLU and polynomial activation functions optimally via relaxations and decomposition methods. It is important to



(a) FLDA (linear)

(b) NFDA (non-linear)

*Figure 5.* Visualization of the optimal transformations for the Yale Face dataset.

emphasize that the proposed methodology solves the problems exactly and involves simple and transparent steps. Our solution is obtained the same way as FLDA applied to the lifted space. Consequently, the solution is projected back to the original space where the optimal network weights are determined. It is important to note that, unlike previous work, our approach obviates the need for non-convex optimization heuristics.

### Acknowledgments

This work was partially supported by an Army Research Office Early Career Award, the National Science Foundation under grants ECCS-2037304, DMS-2134248, and the AI Chip Center for Emerging Smart Systems.

## References

Allen-Zhu, Z. and Li, Y. Backward feature correction: How deep learning performs deep learning. *arXiv preprint arXiv:2001.04413*, 2020.

Bartan, B. and Pilanci, M. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *arXiv preprint arXiv:2101.02429*, 2021a.

Bartan, B. and Pilanci, M. Training quantized neural networks to global optimality via semidefinite programming. In *International Conference on Machine Learning*, pp. 694–704. PMLR, 2021b.

Centeno, T. P. and Lawrence, N. D. Optimising kernel parameters and regularisation coefficients for non-linear discriminant analysis. *Journal of Machine Learning Research*, 7(16):455–491, 2006. URL http://jmlr.org/papers/v7/centeno06a.html.

Chalup, S. K. and Wiklendt, L. Variations of the two-spiral task. *Connect. Sci*, 19(2):183–199, jun 2007. ISSN 0954-0091.

Diamond, S. and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Dorfer, M., Kelz, R., and Widmer, G. Deep linear discriminant analysis. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

Díaz-Vico, D. and Dorronsoro, J. R. Deep least squares fisher discriminant analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 31(8):2752–2763, 2020. doi: 10.1109/TNNLS.2019.2906302.

Edelsbrunner, H., O'Rourke, J., and Seidel, R. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986.

Ergen, T. and Pilanci, M. Convex optimization for shallow neural networks. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 79–83. IEEE, 2019.

Ergen, T. and Pilanci, M. Convex geometry of two-layer relu networks: Implicit autoencoding and interpretable models. In *International Conference on Artificial Intelligence and Statistics*, pp. 4024–4033. PMLR, 2020.

Ergen, T. and Pilanci, M. Convex geometry and duality of over-parameterized neural networks. *Journal of machine learning research*, 2021a.

Ergen, T. and Pilanci, M. Global optimality beyond two layers: Training deep relu networks via convex programs. In *International Conference on Machine Learning*, pp. 2993–3003. PMLR, 2021b.

Ergen, T. and Pilanci, M. Implicit convex regularizers of cnn architectures: Convex optimization of two-and three-layer networks in polynomial time. In *International Conference on Learning Representations (ICLR) 2021*, 2021c.

Ergen, T. and Pilanci, M. Revealing the structure of deep neural networks via convex duality. In *International Conference on Machine Learning*, pp. 3004–3014. PMLR, 2021d.

Ergen, T., Sahiner, A., Ozturkler, B., Pauly, J. M., Mardani, M., and Pilanci, M. Demystifying batch normalization in relu networks: Equivalent convex optimization models and implicit regularization. In *International Conference on Learning Representations*, 2021.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pp. 201–210, 2016.

Gupta, V., Bartan, B., Ergen, T., and Pilanci, M. Exact and relaxed convex formulations for shallow neural autoregressive models. In *International Conference on Acoustics, Speech, and Signal Processing*, 2021.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pp. 41–48, 1999. doi: 10.1109/NNSP.1999.788121.

Mishkin, A., Sahiner, A., and Pilanci, M. Fast convex optimization for two-layer relu networks: Equivalent model classes and cone decompositions. *arXiv preprint arXiv:2202.01331*, 2022.

Pilanci, M. and Ergen, T. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pp. 7695–7705. PMLR, 2020.

Sahiner, A., Ergen, T., Pauly, J. M., and Pilanci, M. Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. In *International Conference on Learning Representations*, 2020.

Sahiner, A., Ergen, T., Ozturkler, B., Bartan, B., Pauly, J., Mardani, M., and Pilanci, M. Hidden convexity of wasserstein gans: Interpretable generative models with closed-form solutions. *International Conference on Learning Representations*, 2021a.

Sahiner, A., Mardani, M., Ozturkler, B., Pilanci, M., and Pauly, J. M. Convex regularization behind neural reconstruction. In *ICLR*, 2021b.

Stuhlsatz, A., Lippel, J., and Zielke, T. Feature extraction with deep neural networks by a generalized discriminant analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 23(4):596–608, 2012. doi: 10.1109/TNNLS.2012.2183645.

van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Wang, Y., Lacotte, J., and Pilanci, M. The hidden convex optimization landscape of regularized two-layer relu networks: an exact characterization of optimal solutions. In *International Conference on Learning Representations*, 2021.

Wu, L., Chunhua Shen, and van den Hengel, A. Deep linear discriminant analysis on fisher networks: A hybrid architecture for person re-identification. *Pattern Recognition*, 65:238–250, 2017. ISSN 0031-3203.

# A. Proofs

This section contains the proofs of the theorems.

*Proof of Theorem 3.1.* Note that we can rewrite the ReLU network as follows

$$f(x) = \sum_{j=1}^{m} (x^T w_j^{(1)})_+ w_j^{(2)} = \sum_{j=1}^{m} (x^T w_j^{(1)})\, \mathbb{1}[x^T w_j^{(1)}] w_j^{(2)}\,. \tag{38}$$

Let us denote the optimal value of the original problem in (13) by $p_1^*$. We now relax the problem by considering the mapping

$$f_{r1}(x) = \sum_{j=1}^{m} (x^T w_j^{(1)})\, \mathbb{1}[x^T h_j \geq 0] w_j^{(2)} \tag{39}$$

where we introduced the variables $h_j \in \mathbb{R}^d$. Note that this is a relaxation since picking $h_j = w_j, j = 1, \ldots, m$ leads to $p_1^*$. Note that the variables of the relaxed problem are $w_j^{(1)}, w_j^{(2)}, h_j, j = 1, \ldots, m$. We will denote the optimal value of the maximization problem with the relaxed mapping by $p_2^*$. It follows that the solution of this problem gives an upper bound for the solution of the original problem $p_2^* \geq p_1^*$.

Next, consider the change of variables $w_j^{(1)} \leftarrow w_j^{(1)} w_j^{(2)}$

$$f_{r1}(x) = \sum_{j=1}^{m} (x^T w_j^{(1)})\, \mathbb{1}[x^T h_j \geq 0]\,. \tag{40}$$

This variable change does not change the optimal value of the optimization problem.

We note that the indicator term makes the problem complicated since $h_j$ is also a variable. Consider all possible values of $\mathbb{1}[x_i^T h_j \geq 0]$ for all the samples $x_i$ of the dataset. Let us represent this as the vector $\mathbb{1}[X h_j \geq 0]$. The number of unique vectors $\mathbb{1}[X h_j \geq 0]$ for all possible $h_j \in \mathbb{R}^d$ is upper bounded by

$$P \leq 2\,\mathrm{rank}(X) \Big( \frac{e(n-1)}{\mathrm{rank}(X)} \Big)^{\mathrm{rank}(X)} \tag{41}$$

for $\mathrm{rank}(X) \leq n$ (Pilanci & Ergen, 2020). Then, the following mapping will lead to a relaxation of the maximization problem:

$$f_{r2}(x) = \sum_{j=1}^{P} (x^T w_j^{(1)})\, \mathbb{1}[x^T \bar{h}_j \geq 0]\,, \tag{42}$$

where $\bar{h}_j$'s are no longer variables, but instead come from the enumeration such that all possible distinct vectors $\mathbb{1}[X\bar{h}_j \geq 0]$ can be obtained. This results in a relaxation since all possible vectors $\mathbb{1}[X\bar{h}_j \geq 0], j = 1, \ldots, P$ include the optimal vectors $\mathbb{1}[X h_j^* \geq 0], j = 1, \ldots, m$. Let us denote the optimal value of the relaxation with the mapping $f_{r2}(x)$ by $p_3^*$. Then, we have $p_3^* \geq p_2^* \geq p_1^*$.

Note that the mapping $f_{r2}(x)$ can be written as

$$f_{r2}(x) = \underbrace{\begin{bmatrix} {w_1^{(1)}}^T & \ldots & {w_P^{(1)}}^T \end{bmatrix}}_{\bar{w}^T} \underbrace{\begin{bmatrix} x\, \mathbb{1}[x^T \bar{h}_1 \geq 0] \\ \vdots \\ x\, \mathbb{1}[x^T \bar{h}_P \geq 0] \end{bmatrix}}_{\bar{x}} \tag{43}$$

where $\bar{x} \in \mathbb{R}^{dP}$ is the lifted input and $\bar{w} \in \mathbb{R}^{dP}$ is the vector that we wish to find. The sample mean for the relaxed mapping

$f_{r2}(x) = \bar{w}^T \bar{x}$ can be computed as follows:

$$\begin{aligned}\beta_k(\theta) &= \mathbb{E}_X[f_{r2}(x)|x \sim \text{ class } k] \\ &= \frac{1}{n_k}\sum_{y_i=k} \bar{w}^T \bar{x}_i = \bar{w}^T \underbrace{\frac{1}{n_k}\sum_{y_i=k} \bar{x}_i}_{\bar{\mu}_k} \\ &= \bar{w}^T \bar{\mu}_k \,. \end{aligned} \tag{44}$$

The sample variance can be computed as follows:

$$\begin{aligned}\sigma_k^2(\theta) &= \mathbb{E}_X[(f_{r2}(x) - \beta_k(\theta))^2|x \sim \text{ class } k] \\ &= \mathbb{E}_X[(\bar{w}^T\bar{x} - \bar{w}^T\bar{\mu}_k)^2|x \sim \text{ class } k] \\ &= \bar{w}^T\, \mathbb{E}_X[(\bar{x} - \bar{\mu}_k)(\bar{x} - \bar{\mu}_k)^T|x \sim \text{ class } k]\bar{w} \\ &= \bar{w}^T \underbrace{\frac{1}{n_k - 1}\sum_{y_i=k}(\bar{x}_i - \bar{\mu}_k)(\bar{x}_i - \bar{\mu}_k)^T}_{\bar{\Sigma}_k} \bar{w} \\ &= \bar{w}^T\bar{\Sigma}_k\bar{w}\,. \end{aligned} \tag{45}$$

Next, we can write the objective as a ratio of two quadratics like in the FLDA method:

$$\bar{w}^* = \arg\max_{\bar{w}} \frac{\bar{w}^T(\bar{\mu}_0 - \bar{\mu}_1)(\bar{\mu}_0 - \bar{\mu}_1)^T\bar{w}}{\bar{w}^T(\bar{\Sigma}_0 + \bar{\Sigma}_1)\bar{w}}\,. \tag{46}$$

It follows that the solution to this problem is given by $\bar{w}^* = c(\bar{\Sigma}_0 + \bar{\Sigma}_1)^{-1}(\bar{\mu}_0 - \bar{\mu}_1)$ where $c \in \mathbb{R}$ is a constant.

Let us denote the blocks of the solution $\bar{w}^*$ as ${w_j^{(1)}}^*$, $j = 1, \ldots, P$. We have already shown through relaxations that the optimal value of (46), denoted $p_3^*$, is an upper bound on the optimal value of the original problem, $p_3^* \geq p_1^*$. We will now show a method for decomposing ${w_j^{(1)}}^*$'s such that the decomposed weights form a feasible two-layer ReLU network. This will enable us to establish a matching lower bound.

In order to find feasible network weights, consider the representation of the ReLU network given in (38) where each neuron is of the form $(x^T w_j^{(1)})\,\mathbb{1}[x^T w_j^{(1)}]w_j^{(2)}$. The main idea behind constructing feasible weights is to find weights $u_j$ and $v_j$ such that the signs of $x_i^T u_j$ and $x_i^T v_j$ are the same as the signs of $x_i^T \bar{h}_j$ for $i = 1, \ldots, n$. Note that this is equivalent to the following two inequalities

$$\begin{aligned}&(2\,\mathbb{1}[x_i^T h_j \geq 0] - 1)(x_i^T u_j) \geq 0, \quad i = 1, \ldots, n, \\ &(2\,\mathbb{1}[x_i^T h_j \geq 0] - 1)(x_i^T v_j) \geq 0, \quad i = 1, \ldots, n\,. \end{aligned} \tag{47}$$

Using the above idea, we now give a linear problem (LP) formulation for computing the desired decomposition

$$\begin{aligned}\text{find } & u_j, v_j \\ \text{s.t. } & u_j - v_j = {w_j^{(1)}}^*,\ \forall j \in [P] \\ & (2\,\mathbb{1}[x_i^T\bar{h}_j \geq 0] - 1)(x_i^T u_j) \geq 0,\ \forall i \in [n], \forall j \in [P] \\ & (2\,\mathbb{1}[x_i^T\bar{h}_j \geq 0] - 1)(x_i^T v_j) \geq 0,\ \forall i \in [n], \forall j \in [P]. \end{aligned} \tag{48}$$

We note that ${w_j^{(1)}}^*$'s are the solution of the Fisher's LDA problem and $\bar{h}_j$'s come from the enumeration. The variables of the above LP are $u_j, v_j \in \mathbb{R}^d$, $j = 1, \ldots, P$.

Let us plug the solution $u_j^*, v_j^*$ of the LP in the relaxed mapping $f_{r2}(x_i)$ in (42):

$$\begin{aligned} f_{r2}(x_i) &= \sum_{j=1}^{P} (x_i^T(u_j^* - v_j^*))\, \mathbb{1}[x_i^T \bar{h}_j \geq 0] \\ &= \sum_{j=1}^{P} x_i^T u_j^* \, \mathbb{1}[x_i^T \bar{h}_j \geq 0] - \sum_{j=1}^{P} x_i^T v_j^* \, \mathbb{1}[x_i^T \bar{h}_j \geq 0] \\ &= \sum_{j=1}^{P} x_i^T u_j^* \, \mathbb{1}[x_i^T u_j^* \geq 0] - \sum_{j=1}^{P} x_i^T v_j^* \, \mathbb{1}[x_i^T v_j^* \geq 0] \\ &= \sum_{j=1}^{P} (x_i^T u_j^*)_+ - \sum_{j=1}^{P} (x_i^T v_j^*)_+ \end{aligned} \tag{49}$$

which holds for $i = 1, \ldots, n$. Recall the original neural network mapping was defined as $f(x_i) = \sum_{j=1}^m (x_i^T w_j^{(1)})_+ w_j^{(2)}$. Note that the last line above is a ReLU neural network with first layer weights $u_1^*, \ldots, u_P^*, v_1^*, \ldots, v_P^*$ and second layer weights $1, \ldots, 1, -1, \ldots, -1$. Let us denote this solution as $\theta_{2P}^*$. The original objective in (13) evaluates to $p_3^*$ at $\theta = \theta_{2P}^*$ due to (49) for $m = 2P$. This implies that $p_3^*$ is a lower bound on $p_1^*$, i.e., $p_3^* \leq p_1^*$. Hence, we have shown that $p_1^* = p_3^*$ for $m = 2P$.

We note that for $m \geq 2P$, the lower bound $p_3^* \leq p_1^*$ still holds since we can pick the extra weights as zero and this would achieve the same objective without the extra weights. Next, when $m \geq P$, the upper bound $p_3^* \geq p_1^*$ still holds since in that case we could combine the neurons with the same indicator vector and this would reduce the total number of neurons down to $P$. Therefore, we obtain that $p_3^* = p_1^*$ for $m \geq 2P$.

Finally, we note that the LP can be stated more compactly. First, recall the definition of the diagonal matrices $D_j := \mathrm{Diag}(\mathbb{1}[X\bar{h}_j \geq 0]) \in \mathbb{R}^{n \times n}$, $j = 1, \ldots, P$. Then, the inequality constraints of the LP in (48) takes the following form

$$\begin{aligned} &(2D_j - I)(Xu_j) \geq 0, \; \forall j \in [P] \\ &(2D_j - I)(Xv_j) \geq 0, \; \forall j \in [P]. \end{aligned} \tag{50}$$

This concludes the proof. □

*Proof of Theorem 3.2.* The transformation by a two-layer neural network with polynomial activation can be written as follows:

$$\begin{aligned} f(x) &= \sum_{j=1}^{m} g(x^T w_j^{(1)}) w_j^{(2)} \\ &= \sum_{j=1}^{m} a(x^T w_j^{(1)})^2 w_j^{(2)} + b(x^T w_j^{(1)}) w_j^{(2)} + c w_j^{(2)} \\ &= \Bigg\langle \underbrace{\begin{bmatrix} a\,\mathrm{vec}(xx^T) \\ bx \\ c \end{bmatrix}}_{\bar{x}}, \underbrace{\begin{bmatrix} \mathrm{vec}\left( \sum_{j=1}^m w_j^{(1)} {w_j^{(1)}}^T w_j^{(2)} \right) \\ \sum_{j=1}^m w_j^{(1)} w_j^{(2)} \\ \sum_{j=1}^m w_j^{(2)} \end{bmatrix}}_{\bar{w}} \Bigg\rangle \end{aligned} \tag{51}$$

where $\langle \cdot, \cdot \rangle$ denotes dot product and $\mathrm{vec}(\cdot)$ is the vectorization operation. Note that above we have defined the lifted input $\bar{x} \in \mathbb{R}^{d^2+d+1}$ and the weight vector $\bar{w} \in \mathbb{R}^{d^2+d+1}$. The above shows that $f(x)$ is equivalent to a linear transformation on the lifted space.

The sample mean and variances can be found using the same derivations that we have used in the ReLU activation (see proof of Theorem 3.1 for more details):

$$\begin{aligned}\beta_k(\theta) &= \mathbb{E}_X[f(x)|x \sim \text{ class } k] \\ &= \bar{w}^T\bar{\mu}_k\,, \qquad (52)\\ \sigma_k^2(\theta) &= \mathbb{E}_X[(f(x)-\beta_k(\theta))^2|x \sim \text{ class } k] \\ &= \bar{w}^T\bar{\Sigma}_k\bar{w}\,. \qquad (53)\end{aligned}$$

Next, we can write the objective as a ratio of two quadratics as in the FLDA method:

$$\bar{w}^* = \arg\max_{\bar{w}} \frac{\bar{w}^T(\bar{\mu}_0-\bar{\mu}_1)(\bar{\mu}_0-\bar{\mu}_1)^T\bar{w}}{\bar{w}^T(\bar{\Sigma}_0+\bar{\Sigma}_1)\bar{w}}\,. \qquad (54)$$

It follows that the solution to this problem is given by $\bar{w}^* = c(\bar{\Sigma}_0+\bar{\Sigma}_1)^{-1}(\bar{\mu}_0-\bar{\mu}_1)$ where $c \in \mathbb{R}$ is a constant.

We now show how to recover neural network weights from the solution $\bar{w}^*$. Construct the matrix $Z^*$ from $\bar{w}^*$ as follows:

$$Z^* = \begin{bmatrix} Z_1^* & Z_2^* \\ Z_2^{*T} & Z_4^* \end{bmatrix} = \begin{bmatrix} \text{vec}^{-1}(\bar{w}^*_{1:d^2}) & \bar{w}^*_{d^2+1:d^2+d} \\ \bar{w}^{*T}_{d^2+1:d^2+d} & \bar{w}^*_{d^2+d+1} \end{bmatrix} \qquad (55)$$

where the notation $w_{i:j}$ is used to point to the entries of $w$ from the $i$'th index to the $j$'th. $\text{vec}^{-1}$ refers to the inverse vectorization which is simply a reshape operation. The blocks of the matrix $Z^* \in \mathbb{R}^{(d+1)\times(d+1)}$ have the following dimensions: $Z_1^* \in \mathbb{R}^{d\times d}, Z_2^* \in \mathbb{R}^d, Z_4^* \in \mathbb{R}$.

Next, compute the eigenvalue decomposition of $Z^*$. Let the eigenvalue decomposition be $Z^* = \sum_j u_j u_j^T \alpha_j$. Let $u_j = \begin{bmatrix} c_j \\ d_j \end{bmatrix} \in \mathbb{R}^{d+1}$ where $c_j \in \mathbb{R}^d$, $d_j \in \mathbb{R}$. The blocks of $Z^*$ can be written in terms of eigenvectors $u_j$'s and eigenvalues $\alpha_j$'s as follows:

$$\begin{aligned} Z^* &= \sum_j \begin{bmatrix} c_j \\ d_j \end{bmatrix} \begin{bmatrix} c_j \\ d_j \end{bmatrix}^T \alpha_j \\ &= \sum_j \begin{bmatrix} c_j/d_j \\ 1 \end{bmatrix} \begin{bmatrix} c_j/d_j \\ 1 \end{bmatrix}^T (\alpha_j d_j^2) \\ &= \begin{bmatrix} \sum_j (c_j/d_j)(c_j/d_j)^T(\alpha_j d_j^2) & \sum_j (c_j/d_j)(\alpha_j d_j^2) \\ \sum_j (c_j/d_j)^T(\alpha_j d_j^2) & \sum_j (\alpha_j d_j^2) \end{bmatrix} \qquad (56)\end{aligned}$$

where we have assumed that $d_j$'s are all nonzero.

The assumption that $d_j$'s are nonzero is without loss of generality. To see this, we will give a probabilistic argument. Consider rewriting $f(x) = \text{tr}(\tilde{x}Z)$ where $\tilde{x}$ is equal to $\bar{x}$ reshaped into a matrix. Sample a random orthogonal matrix $Q$ such that $QQ^T = Q^TQ = I$. Next, note that $f(x) = \text{tr}(\tilde{x}Z) = \text{tr}(Q^T\tilde{x}QQ^TZQ)$. The last entries of the eigenvectors of $Q^TZQ$ will be random and the probability of them being zero is equal to zero. Hence, if any of the $d_j$'s turn out to be zero, then we can consider this random rotation by $Q$.

Note that the above way of decomposing the blocks of $Z^*$ allows us to construct the neural network weights:

$$(w_j^{(1)}, w_j^{(2)}) = (c_j/d_j, \alpha_j d_j^2), \quad j = 1, \ldots, \text{rank}(Z^*). \qquad (57)$$

This concludes the proof.

□

*Proof of Theorem 4.1.* The steps of this proof follow the same outline as the proof for the binary class NFDA.

We begin by relaxing the problem the same way as the binary class case and obtain the following mapping:

$$f_{r1}(x) = \sum_{j=1}^m (x^T w_j^{(1)})\, \mathbb{1}[x^T h_j \geq 0] w_j^{(2)}. \qquad (58)$$

The optimal value $p_2^*$ of this relaxation satisfies $p_2^* \geq p_1^*$. In the next step, we cannot apply the change of variables this time to remove the second layer weights $w_j^{(2)}$ since they are vectors. Instead we will rewrite the second layer weights as a sum of unit vectors $e_k$, $k = 1, \ldots, K-1$ with the appropriate scaling:

$$f_{r1}(x) = \sum_{j=1}^{m} \sum_{k=1}^{K-1} (x^T w_j^{(1)})\, \mathbb{1}[x^T h_j \geq 0] w_{j,k}^{(2)} e_k. \tag{59}$$

where $w_{j,k}^{(2)}$ is the $k$'the entry of $w_j^{(2)}$. Let us define the variables $w_{j,k}^{(1)} \in \mathbb{R}^d$ such that $w_{j,k}^{(1)} = w_j^{(1)} w_{j,k}^{(2)}$. Since we do not restrict $w_{j,k}^{(1)}$ to be the same for different $k$, this also leads to a relaxation of the optimization problem. Plugging these variables in the above mapping, we obtain

$$f_{r1}(x) = \sum_{j=1}^{m} \sum_{k=1}^{K-1} (x^T w_{j,k}^{(1)})\, \mathbb{1}[x^T h_j \geq 0] e_k. \tag{60}$$

Next, we will relax the problem one more time by considering all possible vectors $\mathbb{1}[x^T h_j \geq 0]$. This will result in the following mapping:

$$f_{r2}(x) = \sum_{j=1}^{P} \sum_{k=1}^{K-1} (x^T w_{j,k}^{(1)})\, \mathbb{1}[x^T \bar{h}_j \geq 0] e_k. \tag{61}$$

The optimal value $p_3^*$ for the new relaxation satisfies $p_3^* \geq p_1^*$. We can write $f_{r2}(x)$ as a linear transformation on the lifted inputs as follows:

$$f_{r2}(x) = \underbrace{\begin{bmatrix} {w_{1,1}^{(1)}}^T & {w_{2,1}^{(1)}}^T & \cdots & {w_{P,1}^{(1)}}^T \\ {w_{1,2}^{(1)}}^T & {w_{2,2}^{(1)}}^T & \cdots & {w_{P,2}^{(1)}}^T \\ \vdots & & & \\ {w_{1,K-1}^{(1)}}^T & {w_{2,K-1}^{(1)}}^T & \cdots & {w_{P,K-1}^{(1)}}^T \end{bmatrix}}_{\bar{W}^T} \underbrace{\begin{bmatrix} x\, \mathbb{1}[x^T \bar{h}_1 \geq 0] \\ x\, \mathbb{1}[x^T \bar{h}_2 \geq 0] \\ \vdots \\ x\, \mathbb{1}[x^T \bar{h}_P \geq 0] \end{bmatrix}}_{\bar{x}} \tag{62}$$

where we defined the lifted input $\bar{x} \in \mathbb{R}^{dP}$ and the linear transformation matrix $\bar{W} \in \mathbb{R}^{dP \times (K-1)}$. Next, we note that the solution to the relaxation with the linear mapping $f_{r2}(x) = \bar{W}^T \bar{x}$ is equal to the FLDA solution on the lifted input $\bar{x}$, which is given by the eigenvectors of $\bar{S}_W^{-1} \bar{S}_B$ corresponding to the largest $K-1$ eigenvalues. The scatter matrices $\bar{S}_W$ and $\bar{S}_B$ are defined in the lifted space as follows:

$$\bar{S}_W = \sum_{k=0}^{K-1} \bar{\Sigma}_k, \text{ where} \tag{63}$$

$$\bar{\Sigma}_k = \sum_{y_i = k} (\bar{x}_i - \bar{\mu}_k)(\bar{x}_i - \bar{\mu}_k)^T, \text{and } \bar{\mu}_k = \frac{1}{n_k} \sum_{y_i = k} \bar{x}_i$$

$$\bar{S}_B = \sum_{k=0}^{K-1} n_k (\bar{\mu}_k - \bar{\mu})(\bar{\mu}_k - \bar{\mu})^T \tag{64}$$

where $\bar{\mu} = \frac{1}{n} \sum_{i=1}^{n} \bar{x}_i$. Let us denote the FLDA solution on the lifted input space by $\bar{W}^*$.

We next decompose $\bar{W}^*$ to obtain feasible ReLU network weights by formulating a linear program. Let ${w_{j,k}^{(1)}}^*$ denote the blocks of $\bar{W}^*$. Consider the LP:

$$\begin{aligned} \text{find} \;\; & u_{j,k}, v_{j,k} \\ \text{s.t.} \;\; & u_{j,k} - v_{j,k} = {w_{j,k}^{(1)}}^*, \;\; \forall j \in [P], \forall k \in [K-1] \\ & (2\, \mathbb{1}[x_i^T \bar{h}_j \geq 0] - 1)(x_i^T u_{j,k}) \geq 0, \;\; \forall i \in [n], \forall j \in [P], \forall k \in [K-1] \\ & (2\, \mathbb{1}[x_i^T \bar{h}_j \geq 0] - 1)(x_i^T v_{j,k}) \geq 0, \;\; \forall i \in [n], \forall j \in [P], \forall k \in [K-1]. \end{aligned} \tag{65}$$

Let us denote the solution of the LP by $u_{j,k}^*, v_{j,k}^*$. Plugging the solution in the expression for $f_{r2}(x_i)$ leads to

$$
\begin{aligned}
f_{r2}(x_i) &= \sum_{j=1}^{P} \sum_{k=1}^{K-1} (x_i^T(u_{j,k}^* - v_{j,k}^*))\, \mathbb{1}[x_i^T \bar{h}_j \geq 0] e_k \\
&= \sum_{j=1}^{P} \sum_{k=1}^{K-1} x_i^T u_{j,k}^*\, \mathbb{1}[x_i^T u_{j,k}^* \geq 0] e_k - \sum_{j=1}^{P} \sum_{k=1}^{K-1} x_i^T v_{j,k}^*\, \mathbb{1}[x_i^T v_{j,k}^* \geq 0] e_k \\
&= \sum_{j=1}^{P} \sum_{k=1}^{K-1} (x_i^T u_{j,k}^*)_+ e_k - \sum_{j=1}^{P} \sum_{k=1}^{K-1} (x_i^T v_{j,k}^*)_+ e_k
\end{aligned} \tag{66}
$$

for $i = 1, \ldots, n$. The last line above is a ReLU network with weights $(u_{j,k}^*, e_k)$ and $(u_{j,k}^*, -e_k)$, $j = 1, \ldots, P$ and $k = 1, \ldots, K-1$. The objective evaluated for these weights is equal to $p_3^*$ for $m = 2P(K-1)$ due to (66). Since any ReLU network weights would lead to a lower bound, we have $p_3^* \leq p_1^*$. Hence, we arrive at $p_1^* = p_3^*$ for $m = 2P(K-1)$. The result still holds for $m \geq 2P(K-1)$ due to the same argument that we provided in proof of Theorem 3.1. □

*Proof of Theorem 4.2.* The transformation by a two-layer neural network with polynomial activation can be written as follows:

$$
\begin{aligned}
f(x) &= \sum_{j=1}^{m} g(x^T w_j^{(1)}) w_j^{(2)} \\
&= \sum_{j=1}^{m} a(x^T w_j^{(1)})^2 w_j^{(2)} + b(x^T w_j^{(1)}) w_j^{(2)} + c w_j^{(2)} \\
&= \sum_{j=1}^{m} \sum_{k=1}^{K-1} \Big( a(x^T w_j^{(1)})^2 w_{j,k}^{(2)} + b(x^T w_j^{(1)}) w_{j,k}^{(2)} + c w_{j,k}^{(2)} \Big) e_k
\end{aligned} \tag{67}
$$

where in the last line, we wrote $w_j^{(2)}$'s as a sum of scaled unit vectors.

Next, we will define the variables $w_{j,k}^{(1)} \in \mathbb{R}^d$ and replace the first layer weights with them. Since we do not restrict $w_{j,k}^{(1)}$ to be the same for different $k$, this is in fact a relaxation. Let $f_r(x)$ denote the relaxed transformation. Similarly to the binary class case, we now represent $f_r(x)$ as a linear transformation on the lifted input:

$$
\begin{aligned}
f_r(x) &= \sum_{j=1}^{m} \sum_{k=1}^{K-1} \Big( a(x^T w_{j,k}^{(1)})^2 w_{j,k}^{(2)} + b(x^T w_{j,k}^{(1)}) w_{j,k}^{(2)} + c w_{j,k}^{(2)} \Big) e_k \\
&= \sum_{k=1}^{K-1} \Bigg\langle \underbrace{\begin{bmatrix} a \operatorname{vec}(xx^T) \\ bx \\ c \end{bmatrix}}_{\bar{x}}, \underbrace{\begin{bmatrix} \operatorname{vec}\big( \sum_{j=1}^m w_{j,k}^{(1)} {w_{j,k}^{(1)}}^T w_{j,k}^{(2)} \big) \\ \sum_{j=1}^m w_{j,k}^{(1)} w_{j,k}^{(2)} \\ \sum_{j=1}^m w_{j,k}^{(2)} \end{bmatrix}}_{\bar{w}_k} \Bigg\rangle e_k \\
&= \bar{W}^T \bar{x}
\end{aligned} \tag{68}
$$

where the lifted input is $\bar{x} \in \mathbb{R}^{d^2+d+1}$ and the weight matrix is $\bar{W} \in \mathbb{R}^{(d^2+d+1)\times(K-1)}$. The columns of $\bar{W}$ are denoted $\bar{w}_k \in \mathbb{R}^{d^2+d+1}$.

The scatter matrices are defined the same way as (63). The optimal solution $\bar{W}^*$ is the solution to the FLDA problem on the lifted input space as in the ReLU activation case.

Denote the optimal value for $f_r(x)$ by $p_2^*$ and let the optimal value of the original problem be $p_1^*$ as usual. Then, we have $p_2^* \geq p_1^*$ due to the relaxation.

We now decompose $\bar{W}^*$ into neural network weights which will prove the lower bound and also reveal how the neural network weights are recovered. The method is similar to the binary class case with the difference that we repeat it $K-1$ times for every column $\bar{w}_k^*$ of $\bar{W}^*$. The procedure is as follows: Construct $Z_k^*$ from $\bar{w}_k^*$ using (22). Then, compute the eigenvalue decomposition of $Z_k^*$ where $u_{j,k} = \begin{bmatrix} c_{j,k} \\ d_{j,k} \end{bmatrix}$, $j = 1, \ldots, \operatorname{rank}(Z_k^*)$ are eigenvectors and $\alpha_{j,k}$, $j = 1, \ldots, \operatorname{rank}(Z_k^*)$ are eigenvalues. We repeat this for $k = 1, \ldots, K-1$, which concludes the steps.

Without loss of generality, we assume that $d_{j,k}$'s are nonzero. Then, we construct the neural network weights as $(c_{j,k}/d_{j,k}, \alpha_{j,k} d_{j,k}^2 e_k)$. The total number of neurons is equal to $\sum_{k=1}^{K-1} \operatorname{rank}(Z_k^*)$.

Note that a neural network with the above weights will evaluate to $p_2^*$ when plugged in the original maximization problem and this implies that $p_2^* \leq p_1^*$. Finally, this concludes the proof that $p_2^* = p_1^*$. □