

A RELIEF Based Feature Extraction Algorithm

Yijun Sun*

Dapeng Wu†

Abstract

RELIEF is considered one of the most successful algorithms for assessing the quality of features due to its simplicity and effectiveness. It has been recently proved that RELIEF is an online algorithm that solves a convex optimization problem with a margin-based objective function. Starting from this mathematical interpretation, we propose a novel feature extraction algorithm, referred to as LFE, as a natural generalization of RELIEF. LFE collects discriminant information through local learning, and is solved as an eigenvalue decomposition problem with a closed-form solution. A fast implementation is also derived. Experiments on synthetic and real-world data are presented. The results demonstrate that LFE performs significantly better than other feature extraction algorithms in terms of both computational efficiency and accuracy.

1 Introduction

Feature extraction and selection are two fundamental problems in machine learning research. Not only can their proper design reduce system complexity and processing time, but they can also enhance system performance in many cases. A commonly used method for feature extraction and selection is to pre-multiply a projection matrix to pattern vectors with the aim to optimize a certain criterion function. (By convention, we denote a pattern as a column vector.) For example, in feature selection, the off-diagonal elements of a projection matrix are all set to zero, and the diagonal elements are restricted to take values from the set $\{0, 1\}$. Based on the criterion functions used in search for informative features, feature selection algorithms are traditionally categorized as wrapper and filter methods [1]. In wrapper methods, the performance of a learning algorithm is used to evaluate the goodness of selected feature subsets, while in filter methods criterion functions evaluate feature subsets by their information content, rather than optimizing the performance of any learning algorithm directly. Hence, filter methods are computationally much more efficient, but usually perform worse than wrapper methods. Given a criterion function, feature selection is reduced to a search problem. An exhaustive search is optimum but quickly becomes computationally infeasible with the increase of problem size. Some heuristic combinational

searches, such as forward and/or backward selection [2], are usually employed. These algorithms have shown some successes in practical applications. However, none of them can provide any guarantee of optimality.

The counterpart to feature selection is feature weighting, wherein the diagonal elements of a projection matrix are allowed to take real-valued numbers, instead of from the binary ones. This enables the employment of some well-established optimization techniques and allows for efficient algorithm implementation. Among the existing feature weighting algorithms, RELIEF [3] is considered one of the most successful ones due to its simplicity and effectiveness [4]. It has been recently shown that RELIEF is an online algorithm that solves a convex optimization problem aimed at maximizing a margin-based objective function [5]. The margin is defined based on the one-nearest-neighbor classifier. Compared with filter methods, RELIEF usually performs better due to the performance feedback of a nonlinear classifier when searching for useful features; compared with conventional wrapper methods, by optimizing a convex problem, RELIEF avoids any exhaustive or heuristic combinatorial search, and thus can be implemented efficiently. These two merits explain the success of RELIEF in practical applications.

One major shortcoming of feature weighting and selection is their inability to capture the interaction of correlated features [6]. In some applications, such as face recognition, where to preserve the physical meaning of individual features is not needed, feature extraction is more appropriate than feature weighting and selection. Starting from the mathematical interpretation that RELIEF represents an implementation of a convex optimization problem, we propose a novel feature extraction algorithm, referred to as LFE, as a natural generalization of RELIEF by lifting the diagonal constraints on a pre-multiplied projection matrix. LFE collects discriminant information locally, and is solved as an eigenvalue decomposition problem globally with a closed-form solution. A fast implementation of LFE is also derived. We demonstrate that LFE can be implemented easily with a comparable computational cost to that of principal component analysis. LFE learns a distance metric matrix to approximately optimize the leave-one-out accuracy of a nearest neighbor classifier, and hence LFE has an explicit mechanism to remove irrelevant features. Experiments on synthetic and real-world data are presented. The results demonstrate

*Interdisciplinary Center for Biotechnology Research, University of Florida, Gainesville, FL 32611, USA. Email: sunyijun@biotech.ufl.edu

†Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. Email: wu@ece.ufl.edu

that LFE performs significantly better than other feature extraction algorithms in terms of computational efficiency and accuracy.

The remainder of the paper is organized as follows. In Section 2, we provide a mathematical interpretation of RELIEF. In Section 3, we propose a new feature extraction algorithm, referred to as LFE. A fast implementation of LFE is derived and computational complexity analysis is presented. In Section 4, some related work is briefly reviewed. In Section 5, numerical experiments are performed. We finally conclude this paper in Section 6.

2 Mathematical Interpretation of RELIEF

We first present a brief review of RELIEF. Let $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^I \times \{\pm 1\}$ be a training dataset, where \mathbf{x}_n is the n -th data sample and y_n is its corresponding class label. The basic idea of RELIEF is to iteratively estimate feature weights according to their ability to discriminate between neighboring patterns. In each iteration, a pattern \mathbf{x} is randomly selected and then two nearest neighbors of \mathbf{x} are found, one from the same class (termed the *nearest hit* or NH) and the other from the opposite class (termed the *nearest miss* or NM). The weight of the i -th feature is then updated as: $w_i = w_i + |\mathbf{x}^{(i)} - \text{NM}^{(i)}(\mathbf{x})| - |\mathbf{x}^{(i)} - \text{NH}^{(i)}(\mathbf{x})|$, for $1 \leq i \leq I$.

We provide a mathematical interpretation for the seemingly heuristic RELIEF algorithm. Following the margin definition in [7], we define the margin for a pattern \mathbf{x}_n as $\rho_n = d(\mathbf{x}_n - \text{NM}(\mathbf{x}_n)) - d(\mathbf{x}_n - \text{NH}(\mathbf{x}_n))$, where $\text{NM}(\mathbf{x}_n)$ and $\text{NH}(\mathbf{x}_n)$ are the nearest miss and hit of \mathbf{x}_n , respectively, and $d(\cdot)$ is a distance function. For the purpose of this paper, we define $d(\mathbf{x}) = \sum_i |x_i|$, which is consistent with the distance function used in the original RELIEF algorithm. Other distance functions may also be used. Note that $\rho_n > 0$ if only if \mathbf{x}_n is correctly classified by the one-nearest-neighbor classifier. Ideally, we may want to scale each feature so that the leave-one-out error $\sum_{n=1}^N \mathbf{I}(\rho_n(\mathbf{w}) < 0)$ is minimized, where $\mathbf{I}(\cdot)$ is the indicator function and $\rho_n(\mathbf{w})$ is the margin of \mathbf{x}_n computed with respect to \mathbf{w} . Since the indicator function is not differentiable, we instead use a linear utility function so that the averaged margin in a weighted feature space is maximized:

$$(2.1) \quad \begin{aligned} \max_{\mathbf{w}} \quad & \sum_{n=1}^N \rho_n(\mathbf{w}) = \sum_{n=1}^N \left(\sum_{i=1}^I w_i |\mathbf{x}_n^{(i)} - \text{NM}^{(i)}(\mathbf{x}_n)| \right. \\ & \left. - \sum_{i=1}^I w_i |\mathbf{x}_n^{(i)} - \text{NH}^{(i)}(\mathbf{x}_n)| \right), \\ \text{s.t.} \quad & \|\mathbf{w}\|_2^2 = 1, \mathbf{w} \geq 0, \end{aligned}$$

where the constraint $\|\mathbf{w}\|_2^2 = 1$ prevents the maximization from increasing without bound and $\mathbf{w} \geq 0$ ensures that the learned weight vector induces a distance measure. By

defining $\mathbf{z} = \sum_{n=1}^N (|\mathbf{x}_n - \text{NM}(\mathbf{x}_n)| - |\mathbf{x}_n - \text{NH}(\mathbf{x}_n)|)$, where $|\cdot|$ is the point-wise absolute operator, Eq. (2.1) can be simplified to read

$$(2.2) \quad \max_{\mathbf{w}} \mathbf{w}^T \mathbf{z}, \text{ s.t. } \|\mathbf{w}\|_2^2 = 1, \mathbf{w} \geq 0.$$

The Lagrangian of Eq. (2.2) is $L = -\mathbf{w}^T \mathbf{z} + \lambda(\|\mathbf{w}\|_2^2 - 1) + \sum_{i=1}^I \theta_i(-w_i)$, where λ and $\theta \geq 0$ are Lagrangian multipliers. Taking the derivative of L with respect to \mathbf{w} and setting it to zero yields $\partial L / \partial \mathbf{w} = -\mathbf{z} + 2\lambda \mathbf{w} - \theta = 0$ and $\mathbf{w} = (\mathbf{z} + \theta) / 2\lambda$. Below, we derive a closed-form solution for \mathbf{w} . We first prove that $\lambda > 0$. Suppose there exists $z_i > 0$.¹ Then $z_i + \theta_i > 0$. If $\lambda < 0$, then we would have $w_i < 0$, which contradicts the constraint $\mathbf{w} \geq 0$. By using the Karush-Kuhn-Tucker condition [8], namely $\sum_i \theta_i w_i = 0$, it is easy to verify the following three cases: (1) $z_i = 0 \Rightarrow \theta_i = 0$ and $w_i = 0$; (2) $z_i > 0 \Rightarrow z_i + \theta_i > 0 \Rightarrow w_i > 0 \Rightarrow \theta_i = 0$; and (3) $z_i < 0 \Rightarrow \theta_i > 0 \Rightarrow w_i = 0 \Rightarrow z_i = -\theta_i$. It follows that the optimum solution can be calculated in a closed form as $\mathbf{w} = (\mathbf{z}^+) / \|\mathbf{z}^+\|_2$, where $(\mathbf{z}^+) = [\max(z_1, 0), \dots, \max(z_I, 0)]^T$.

By comparing the expression of \mathbf{w} with the weight update rule of RELIEF, we conclude that RELIEF is an online algorithm to solve the optimization scheme Eq. (2.1). This is true except when $w_i = 0$ for $z_i \leq 0$, which usually corresponds to irrelevant features.

RELIEF maximizing the averaged margin was first observed in [7], but no mathematical proof was provided. From our analysis, we find that RELIEF is a feature weighting algorithm that utilizes the performance of a nonlinear classifier in searching for useful features, yet results in a simple convex problem with a closed-form solution. This clearly explains the simplicity and effectiveness of RELIEF in real applications.

3 Learning Distance Metric Matrix

A natural extension of RELIEF is to use a full distance metric matrix instead of a diagonal one. Let us consider the following optimization problem:

$$(3.3) \quad \begin{aligned} \max_{\mathbf{W}} \quad & \sum_{n=1}^N \rho_n(\mathbf{W}) = \sum_{n=1}^N \mathbf{m}_n^T \mathbf{W} \mathbf{m}_n - \sum_{n=1}^N \mathbf{h}_n^T \mathbf{W} \mathbf{h}_n, \\ \text{s.t.} \quad & \|\mathbf{W}\|_F^2 = 1, \mathbf{W} \geq 0, \end{aligned}$$

where $\mathbf{m}_n = \mathbf{x}_n - \text{NM}(\mathbf{x}_n)$, $\mathbf{h}_n = \mathbf{x}_n - \text{NH}(\mathbf{x}_n)$, and $\|\mathbf{W}\|_F$ is the Frobenius norm of \mathbf{W} , defined as $\sqrt{\sum_{i,j} w_{i,j}^2} = \sqrt{\sum_i \lambda_i^2}$, with $\{\lambda_i\}_{i=1}^I$ being the set of

¹This assumption is very weak since if it does not hold it simply says that on average the distance between a pattern and its nearest miss is larger than the distance of the pattern from its nearest hit, which is very rare in real applications.

eigenvalues of \mathbf{W} . Eq. (3.3) has the same physical meaning as Eq. (2.1).

The direct optimization of Eq. (3.3) is computationally arduous. We therefore derive a closed-form solution. Some organization is merited. First, since \mathbf{W} is a distance metric matrix, it is symmetric and positive-semidefinite, and thus can be written as $\mathbf{W} = \mathbf{A}\mathbf{A}^T$, where $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_I]$. We further assume that $\{\mathbf{a}_i\}_{i=1}^I$ are pairwise orthogonal, that is, $\langle \mathbf{a}_i, \mathbf{a}_j \rangle = 0$, for any $i \neq j$. This can be easily done through the eigenvalue decomposition of \mathbf{W} as follows:

$$(3.4) \quad \begin{aligned} \mathbf{W} &= \Phi \Lambda \Phi^T = \Phi \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} \Phi, \\ &= [\sqrt{\lambda_1} \phi_1, \dots, \sqrt{\lambda_I} \phi_I] [\sqrt{\lambda_1} \phi_1, \dots, \sqrt{\lambda_I} \phi_I]^T, \end{aligned}$$

Where $\{\phi_i\}_{i=1}^I$ and $\{\lambda_i\}_{i=1}^I$ are the eigenvectors and eigenvalues of \mathbf{W} , respectively. Let $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_I] = [\sqrt{\lambda_1} \phi_1, \dots, \sqrt{\lambda_I} \phi_I]$. The task then is to solve for the set $\{\mathbf{a}_i\}_{i=1}^I$. The first term of the objective function in Eq. (3.3) can be rearranged:

$$(3.5) \quad \begin{aligned} &\sum_{n=1}^N \mathbf{m}_n^T \mathbf{W} \mathbf{m}_n \\ &= \text{tr} \left(\mathbf{W} \sum_{n=1}^N \mathbf{m}_n \mathbf{m}_n^T \right), \\ &= \text{tr} (\mathbf{W} \Sigma_m), \quad (\Sigma_m \triangleq \sum_{n=1}^N \mathbf{m}_n \mathbf{m}_n^T) \\ &= \text{tr} \left(\Sigma_m \sum_{i=1}^I \mathbf{a}_i \mathbf{a}_i^T \right) = \sum_{i=1}^I \mathbf{a}_i^T \Sigma_m \mathbf{a}_i. \end{aligned}$$

As a result, the objective function of Eq. (3.3) can be expressed as follows:

$$(3.6) \quad \begin{aligned} &\sum_{n=1}^N \mathbf{m}_n^T \mathbf{W} \mathbf{m}_n - \sum_{n=1}^N \mathbf{h}_n^T \mathbf{W} \mathbf{h}_n, \\ &= \sum_{i=1}^I \mathbf{a}_i^T (\Sigma_m - \Sigma_h) \mathbf{a}_i. \quad (\Sigma_h \triangleq \sum_{n=1}^N \mathbf{h}_n \mathbf{h}_n^T) \end{aligned}$$

Similarly, the optimization constraint can be expressed as a function of the column vectors $\{\mathbf{a}_i\}_{i=1}^I$:

$$(3.7) \quad \begin{aligned} \|\mathbf{W}\|_F^2 &= \sum_{s,t} w_{s,t}^2 = \sum_{s,t} \left(\sum_i \mathbf{a}_i^{(s)} \mathbf{a}_i^{(t)} \right)^2, \\ &= \sum_{i,j} \left(\sum_s \mathbf{a}_i^{(s)} \mathbf{a}_j^{(s)} \right)^2 = \sum_{i,j} (\mathbf{a}_i^T \mathbf{a}_j)^2 \end{aligned}$$

An equivalent optimization problem of Eq. (3.3) can thus be formulated:

$$(3.8) \quad \begin{aligned} \max_{\{\mathbf{a}_i\}_{i=1}^I} &\sum_{i=1}^I \mathbf{a}_i^T (\Sigma_m - \Sigma_h) \mathbf{a}_i, \\ \text{s.t.} &\sum_{i,j} (\mathbf{a}_i^T \mathbf{a}_j)^2 = \sum_i (\mathbf{a}_i^T \mathbf{a}_i)^2 = 1. \end{aligned}$$

The equality in the constraint is due to the orthogonality of $\{\mathbf{a}_i\}_{i=1}^I$. Though we do not list the orthogonality constraint explicitly in the optimization, we will later see that this constraint is automatically fulfilled. The Lagrangian of Eq. (3.8) is:

$$(3.9) \quad L = - \sum_{i=1}^I \mathbf{a}_i^T (\Sigma_m - \Sigma_h) \mathbf{a}_i + \lambda \left(\sum_{i=1}^I (\mathbf{a}_i^T \mathbf{a}_i)^2 - 1 \right).$$

Define $\Sigma_{mh} \triangleq \Sigma_m - \Sigma_h$ to simplify the notation. Taking the derivative of L with respect to \mathbf{a}_i and setting it to zero yields:

$$(3.10) \quad \begin{aligned} \frac{\partial L}{\partial \mathbf{a}_i} &= -2\Sigma_{mh} \mathbf{a}_i + 4\lambda \mathbf{a}_i^T \mathbf{a}_i \mathbf{a}_i = 0, \\ \Rightarrow \Sigma_{mh} \mathbf{a}_i / \|\mathbf{a}_i\|_2 &= 2\lambda \|\mathbf{a}_i\|_2^2 \mathbf{a}_i / \|\mathbf{a}_i\|_2, \\ \Rightarrow \Sigma_{mh} \bar{\mathbf{a}}_i &= 2\lambda \|\mathbf{a}_i\|_2^2 \bar{\mathbf{a}}_i, \quad (\text{Assume } \|\mathbf{a}_i\|_2 \neq 0.) \end{aligned}$$

where $\bar{\mathbf{a}}_i = \mathbf{a}_i / \|\mathbf{a}_i\|_2$. If we define $\sigma_i = 2\lambda \|\mathbf{a}_i\|_2^2$, then $\Sigma_{mh} \bar{\mathbf{a}}_i = \sigma_i \bar{\mathbf{a}}_i$ is the eigen-system of Σ_{mh} . Note that the above equation also holds if $\mathbf{a}_i = \mathbf{0}$. Therefore, the solutions $\{\mathbf{a}_i\}_{i=1}^I$ can be written as

$$(3.11) \quad \mathbf{a}_i = \sqrt{\beta_i} \bar{\mathbf{a}}_i, \quad \beta_i \geq 0, \quad 1 \leq i \leq I.$$

Substituting Eq. (3.11) into Eq. (3.8), we obtain an optimization problem of the following form:

$$(3.12) \quad \begin{aligned} \max_{\beta} &\beta^T \sigma, \\ \text{s.t.} &\|\beta\|_2^2 = 1, \beta \geq 0, \end{aligned}$$

This problem is identical to the one expressed by Eq. (2.2). The conclusion drawn in the last section can be directly applied here: for $\{i | \sigma_i > 0, 1 \leq i \leq I\}$,

$$(3.13) \quad \beta_i = \sigma_i / \sqrt{\sum_{\{j | \sigma_j > 0\}} \sigma_j^2},$$

$$(3.14) \quad \mathbf{a}_i = \bar{\mathbf{a}}_i \sqrt{\sigma_i / \sqrt{\sum_{\{j | \sigma_j > 0\}} \sigma_j^2}},$$

and for $\{i | \sigma_i \leq 0, 1 \leq i \leq I\}$,

$$(3.15) \quad \beta_i = 0 \quad \text{and} \quad \mathbf{a}_i = \mathbf{0}.$$

Note that the orthogonality constraint of $\{\mathbf{a}_i\}_{i=1}^I$ is automatically fulfilled.

3.1 Feature Extraction One direct application of Eq. (3.3) is for feature extraction. The physical meaning of Eq. (3.3) is that we project the data onto a subspace spanned by $\{\mathbf{a}_i\}$ so that the average margin in the projected subspace is maximized. Recall that principal component analysis (PCA) determines the projection directions so that the mean-squared reconstruction error is minimized. The solutions are the eigenvectors of a covariance matrix corresponding to the largest eigenvalues, and the reconstruction error is simply the sum of the discarded eigenvalues. Similarly, we can regard the objective function of Eq. (3.3) as the total discriminant power, which can be shown to be $\sqrt{\sum_{\{i | \sigma_i > 0\}} \sigma_i^2}$ by plugging Eqs. (3.13), (3.14) and (3.15) into Eq. (3.12). Forming a projection matrix \mathbf{A} by only using $\{\mathbf{a}_i\}$ corresponding to the largest eigenvalues of Σ_{mh} , the suffered

discriminant power loss is proportional to the sum of the squared positive eigenvalues that are discarded. More precisely, let $\sigma_1 \geq \dots \geq \sigma_T > 0 > \sigma_{T+1} \geq \dots \geq \sigma_I$. By using the first t eigenvectors with $t < T$, the discriminant power loss is $\sum_{i=t+1}^T \sigma_i^2 / \sqrt{\sum_{\{j|\sigma_j>0\}} \sigma_j^2}$. We name the newly proposed algorithm as Local Feature Extraction, or LFE for short, since the discriminant information is collected through local learning.

LFE has several nice properties. First, since local learning is a highly nonlinear process, LFE is capable of extracting nonlinear discriminant information. Second, the implementation of LFE is very easy. We will later show that the computational cost of LFE is of the same order as that of PCA. Third, LFE has an explicit mechanism to eliminate irrelevant features. This property is inherited from RELIEF that approximately optimizes the leave-one-out accuracy of the nearest neighbor classifier (see Eq. (2.1)). It can be shown that the diagonal elements of matrix \mathbf{W} approximately equal to the weights learned in RELIEF. That is, if there exists an irrelevant feature, then the corresponding diagonal term of \mathbf{W} will take a small value near zero. This property is experimentally verified in our experiment (see Fig. 3).

3.2 Fast Implementation of LFE The major computation complexity of LFE comes from the eigenvalue decomposition of Σ_{mh} . Therefore, LFE has the same computational complexity as PCA. However, in many practical applications, such as face recognition, the data dimensionality is much larger than the number of available training samples. Both PCA and LFE can be implemented efficiently since the information is encoded in a much smaller subspace. The derivation of the fast PCA implementation is straightforward, and herein we only present a fast implementation for LFE.

We first denote $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$. For notational simplicity, we assume that the data has been centered, that is, $\sum_{n=1}^N \mathbf{x}_n = \mathbf{0}$. Let $\mathbf{X}\mathbf{X}^T = \Psi\Delta\Psi^T$ be the eigenvalue decomposition of the scatter matrix $\mathbf{X}\mathbf{X}^T$, where the eigenvalues in Δ are sorted in a decreasing order. The pattern \mathbf{x} can be expressed as $\mathbf{x} = \Psi_1\mathbf{y}$, where Ψ_1 contains the eigenvectors corresponding to the first N largest eigenvalues of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{y} = \Psi_1^T\mathbf{x}$. Then, $\Sigma_{mh}\bar{\mathbf{a}}_i = \sigma_i\bar{\mathbf{a}}_i$ can be written as

$$(3.16) \quad \Psi \begin{bmatrix} \Sigma_{mhy} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \Psi^T \bar{\mathbf{a}}_i = \sigma_i \bar{\mathbf{a}}_i,$$

where Σ_{mhy} is the same as Σ_{mh} , but is computed with \mathbf{y} . Defining $\tilde{\mathbf{a}}_i = \Psi^T \bar{\mathbf{a}}_i$ and pre-multiplying Ψ^T to both sides of Eq. (3.16) yield

$$(3.17) \quad \begin{bmatrix} \Sigma_{mhy} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{a}}_{i,1} \\ \tilde{\mathbf{a}}_{i,2} \end{bmatrix} = \sigma_i \begin{bmatrix} \tilde{\mathbf{a}}_{i,1} \\ \tilde{\mathbf{a}}_{i,2} \end{bmatrix}.$$

It immediately follows that $\Sigma_{mhy}\tilde{\mathbf{a}}_{i,1} = \sigma_i\tilde{\mathbf{a}}_{i,1}$ and $\tilde{\mathbf{a}}_{i,2} = \mathbf{0}$. Note that Σ_{mhy} is an $N \times N$ matrix that has much smaller dimensions than Σ_{mh} , and both matrices share the same set of non-zero eigenvalues. With $\tilde{\mathbf{a}}_i$ computed, $\bar{\mathbf{a}}_i = \Psi\tilde{\mathbf{a}}_i = \Psi_1\tilde{\mathbf{a}}_{i,1}$.

We now discuss the computational complexity of LFE and PCA. If $N > I$, the complexity of LFE and PCA is $\mathcal{O}(NI^2) + \mathcal{O}(I^3)$, where the first term is for the computation of a scatter matrix and the second term is for the eigenvalue decomposition. If $N < I$, the complexity of PCA and LFE are $\mathcal{O}(IN^2) + \mathcal{O}(N^3)$ and $\mathcal{O}(IN^2) + 2\mathcal{O}(N^3)$, respectively. In both cases, LFE has a comparable computational cost to PCA.

4 Related Work

We briefly review some feature extraction algorithms that we will compare with LFE in our experiments. PCA is probably one of the most commonly used algorithms in the literature for feature extraction. This is largely because it has a clear physical meaning and can be implemented easily. One major drawback of PCA, however, is that the top ranked principal components are not necessarily the ones containing the most discriminant information. We do not consider other PCA-type algorithms (e.g. kernel PCA) in the experiment because these algorithms, though performing better than PCA in detecting nonlinear embedding, all ignore class label information in a learning process and suffer from the same problems as PCA. Linear discriminant analysis (LDA) tries to overcome the problem of PCA by projecting data onto a subspace so that the ratio between the determinant or trace of the between-class scatter matrix and that of the within-class scatter matrix in the projected subspace is maximized. However, LDA can at most generate $C - 1$ features, where C is the number of classes. Moreover, it is well known that LDA is not robust when the feature dimensionality is high compared to the sample size.

In [9], a local learning based feature extraction algorithm, referred to as DANN, is proposed. DANN projects data onto a subspace spanned by the eigenvectors corresponding to the largest eigenvalues of an average local between-class scatter matrix. Favorable results have been reported in [9], indicating that DANN has the capability to extract discriminant information and remove irrelevant features. However, the objective function optimized by DANN is not directly related to the performance of any specific learning algorithm. Hence, DANN can be regarded as a filter method in the context of feature extraction. We will later see in the experiment that the performance of DANN relies heavily on the correct estimation of the number of extracted features.

More recently, a distance metric learning algorithm named neighborhood component analysis (NCA) is proposed in [10]. The basic idea of NCA is to estimate a dis-

Table 1: Data Summary

database	training	test	features
<i>banana</i>	400	4900	2
<i>twonorm</i>	400	7000	20
<i>waveform</i>	400	4600	21
<i>ringnorm</i>	400	7000	20
<i>splice</i>	1000	2175	60
<i>thyroid</i>	140	75	5
<i>face</i>	500	1456	5100

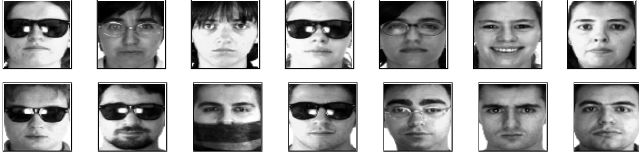


Figure 1: Face dataset: female (top row) and male (bottom row).

tance metric matrix (or equivalently a projection matrix) to minimize the error probability under stochastic neighborhood assignment. Our work shares the similar basis to NCA in the sense that both learn a projection matrix based on local information. However, unlike LFE, NCA is a non-convex optimization problem, and its implementation completely relies on gradient descent. Hence, NCA does not scale well to a classification problem with a large feature dimensionality. For example, in the face recognition problem that we consider in our experiment, an image is of size 85×60 . To learn a full metric matrix, NCA essentially performs gradient-descent search in a space of size 5100×5100 , which is computationally infeasible.

Another related algorithm is LMNN [11]. Unlike NCA, LMNN is posed as a convex problem, and thus the reach of the global solution is guaranteed. However, a special optimization solver is needed for efficient implementation. It is reported that LMMN performs similarly to NCA [11].

5 Experiments

We conduct some experiments to demonstrate the effectiveness of LFE on six UCI datasets [12] and the AR face recognition (*face*) dataset [13]. The six UCI datasets include *banana*, *twonorm*, *waveform*, *ringnorm*, *diabetics*, *thyroid*, and *splice*. For the *face* dataset, the task is to classify female against male (see Fig. 1 for some sample images). The data information is summarized in Table 1. Note that in the *face* dataset the feature dimensionality significantly outnumbers the training samples. It is computationally infeasible to directly apply NCA in the original feature space. We therefore first perform PCA that projects data onto its leading 50 principal components and then perform NCA.

database	NCA	DANN	LFE	PCA
<i>banana</i>	12.6	3.6	3.3	3.0
<i>twonorm</i>	41.6	4.7	4.8	4.7
<i>waveform</i>	38.8	4.4	4.3	4.2
<i>ringnorm</i>	39.4	4.7	4.7	4.7
<i>splice</i>	383.6	8.4	8.4	4.8
<i>thyroid</i>	4.9	3.2	3.3	3.2
<i>face</i>	185.7	24.9	23.8	8.6

Table 3: CPU time (second) per run (Pentium4 2.80GHz with 2.00GB RAM)

5.1 Experiments on UCI Datasets We first compare LFE with RELIEF, PCA, NCA, and DANN on the UCI datasets. In order to demonstrate that LFE has an explicit mechanism to eliminate irrelevant features, we add 10 independent Gaussian distributed irrelevant features to each pattern. We compare RELIEF and LFE to illustrate when and how LFE can improve classification performance of RELIEF. It should be emphasized that feature weighting and extraction are used in different scenarios. In the experiment we actually use RELIEF-F [14], which uses M , instead of just one, nearest hits and misses in computing sample margins to ensure greater robustness of the algorithm against noise. The value of M is found through 10-fold cross-validation on training data. For the same reason, we search for multiple nearest neighbors in LFE.

KNN is used to estimate the classification errors of each algorithms for two reasons. First, KNN is a very simple yet effective classifier. Given a proper distance metric used to identify nearest neighbors, KNN often yields competitive results to some advanced classification algorithms. For this reason, many distance metric learning algorithms (e.g., [10, 11]) are specifically designed to improve the performance of KNN. Second, using KNN makes our experiment computationally feasible. KNN is certainly not an optimal classifier for each dataset. However, the focus of this paper is not on the optimal classifier design but on feature extraction. KNN provides us with a platform where we can compare different feature-extraction algorithms with a reasonable computational cost. The number of the nearest neighbors is estimated through 10-fold cross-validation using training data. In Section 5.2, we perform an experiment on using LFE to improve the performance of SVM.

To eliminate statistical variations, each algorithm is run 20 times for each dataset. In each run, a dataset is randomly partitioned into training and testing, and 10 irrelevant features are added. The testing errors of DANN, NCA, LFE and PCA as a function of the number of extracted features for each dataset, averaged over 20 runs, are plotted in Fig. 2. In Table 2, the testing errors and the standard deviations (STDs) are reported. For RELIEF, after finding

Table 2: The testing errors and STDs (%) on six UCI datasets. When the difference between weighted and unweighted approaches (KNN performed on the original data) are more than three STDs, the results are boldfaced. $0.01^+(0.01^-)$ means LFE performs better (worse) than RELIEF or NCA at 0.01 p-value level.

Dataset	KNN	RELIEF	DANN	NCA	LFE	p-value (LFE/RELIEF)	p-value (LFE/NCA)
<i>banana</i>	37.1(2.1)	12.3(0.7)	23.0(4.4)	25.4(4.4)	18.2(2.5)	$< 0.001^-$	$< 0.001^+$
<i>splice</i>	26.6(1.8)	11.4(0.9)	14.1(0.7)	13.9(0.4)	12.0(0.7)	0.02^-	0.002^+
<i>waveform</i>	12.6(0.7)	10.8(0.5)	11.1(0.8)	10.9(0.5)	9.8 (0.6)	$< 0.001^+$	$< 0.001^+$
<i>ringnorm</i>	39.2(1.3)	31.3(2.8)	24.1(2.7)	26.8(1.2)	22.0(1.3)	$< 0.001^+$	$< 0.001^+$
<i>twonorm</i>	3.9(0.3)	3.5(0.4)	2.9(0.3)	3.8(0.5)	2.6(0.2)	$< 0.001^+$	$< 0.001^+$
<i>thyroid</i>	17.0(3.0)	9.5(3.0)	8.8(3.4)	7.0(2.6)	6.2(2.8)	0.001^+	0.36^+

feature weights, we first remove the features corresponding to the negative weights, and then perform classification on the weighted feature space. For both DANN and LFE, the optimum number of features used in KNN is estimated through cross-validation. For NCA, due to the computational reason, we simply record the minimum value of the average error of each dataset. For a rigorous comparison of LFE with RELIEF and NCA, a Student’s paired two-tailed t-test is also performed. The p-value of the t-test reported in Table 2 represents the probability that two sets of compared results come from distributions with equal means. A p-value of 0.05 is considered statistically significant. For comparison, the classification errors of KNN performed on the original data are also reported. From these experimental results, we arrive at the following observations:

(1) From Fig. 2, we observe that with respect to classification errors and the effectiveness in reducing data dimensionality, LFE performs the best, DANN and NCA the second, and PCA the worst. The performance of DANN is conditioned heavily on the correct estimation of the number of extracted features, whereas both LFE and NCA are very robust against this parameter, which makes model selection easy in real applications. Compared with NCA, LFE performs significantly better on five out of six datasets (p-value < 0.01 , Table 2).

(2) In Table 3, we report the CPU time per run of PCA, NCA and LFE for each dataset. NCA is much more computationally expensive than DANN and LFE, while the latter two has a comparable computational cost to PCA.

(3) Both RELIEF and LFE can significantly improve the performance of KNN performed on the original datasets. However, the performance of feature weighting and extraction largely depends on the characteristic of the studied datasets. We particularly examine four datasets that have significant performance differences between RELIEF and LFE. As one moves from *banana* to *splice*, *waveform*, and *ringnorm*, and then to *twonorm*, the degree of correlation increases significantly, manifested in the learned distance metric matrices \mathbf{W} plotted in Fig. 3. As a result, LFE performs

worse than RELIEF on the first two datasets, but significantly better on the last three ones (p-value < 0.001). From this experiment, we conclude that when there exists feature interaction, LFE has a clear advantage over RELIEF.

(4) An important feature of LFE, compared to other feature extraction algorithms such as PCA and DANN, is that LFE has an *explicit* mechanism to eliminate irrelevant features, a nice property inherited from RELIEF. We observe in the plotted distance metric matrices (Fig. 3) that the lower right corner (the last 10 rows and columns) that corresponds to artificially added irrelevant features takes small values near zero. This explains why as to the error curves of LFE becomes flatten after reaching the minimum errors in almost all datasets (Fig. 2).

5.2 Experiments on Face Dataset We finally compare LFE with DANN, NCA and PCA on the *face* dataset. The testing results of DANN, PCA and LFE as a function of the top 50 extracted features, averaged over 20 runs, are plotted in Fig. 4. Due to the high computational cost associated with NCA, we only perform NCA on four levels of feature numbers (i.e., 10, 20, 30, 40, 50). The results are quite consistent with those of the UCI datasets. LFE reaches the minimum classification errors around 20 features, and DANN, PCA and NCA perform worse than LFE for nearly all feature numbers. With 20 features, the classification errors (STDs) of LFE, NCA, DANN, and PCA are 8.0 (1.0)%, 11.8(1.8)%, 11.4 (0.7)% and 13.5 (1.3)%, respectively. Our algorithm performs significantly better than others (p-value < 0.01). A similar experiment is preformed in [7] to compare different feature selection algorithms. With 100 features, RELIEF achieves about 80% accuracy, which is significantly worse than that of LFE with only 20 features.

In Table 3, the CPU times of LFE, DANN, NCA and LFE are listed. Both DANN and LFE are much computationally efficient than NCA, though NCA is only performed on a 50-dimensional PCA subspace.

We have so far shown that LFE can significantly improve the performance of KNN. In fact, any classification

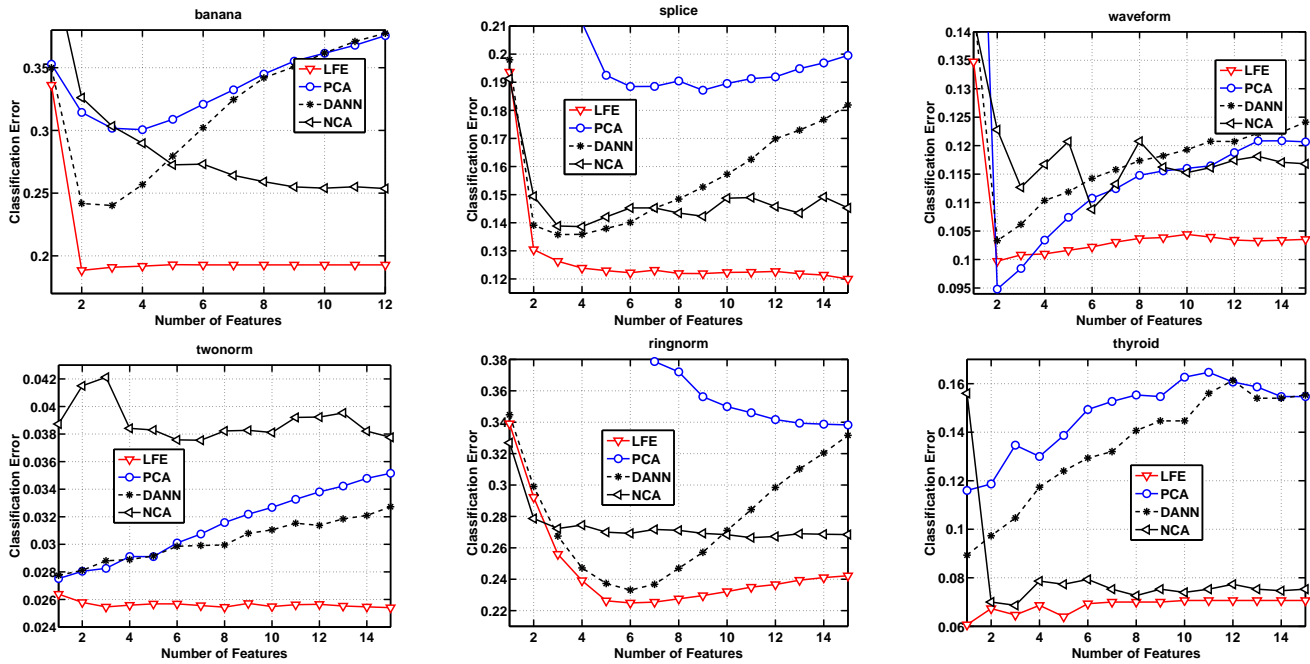


Figure 2: Classification errors of PCA, LFE, NCA and DANN on the UCI datasets. In terms of both classification errors and the effectiveness in reducing data dimensionality, LFE performs the best, DANN and NCA the second, and PCA the worst.

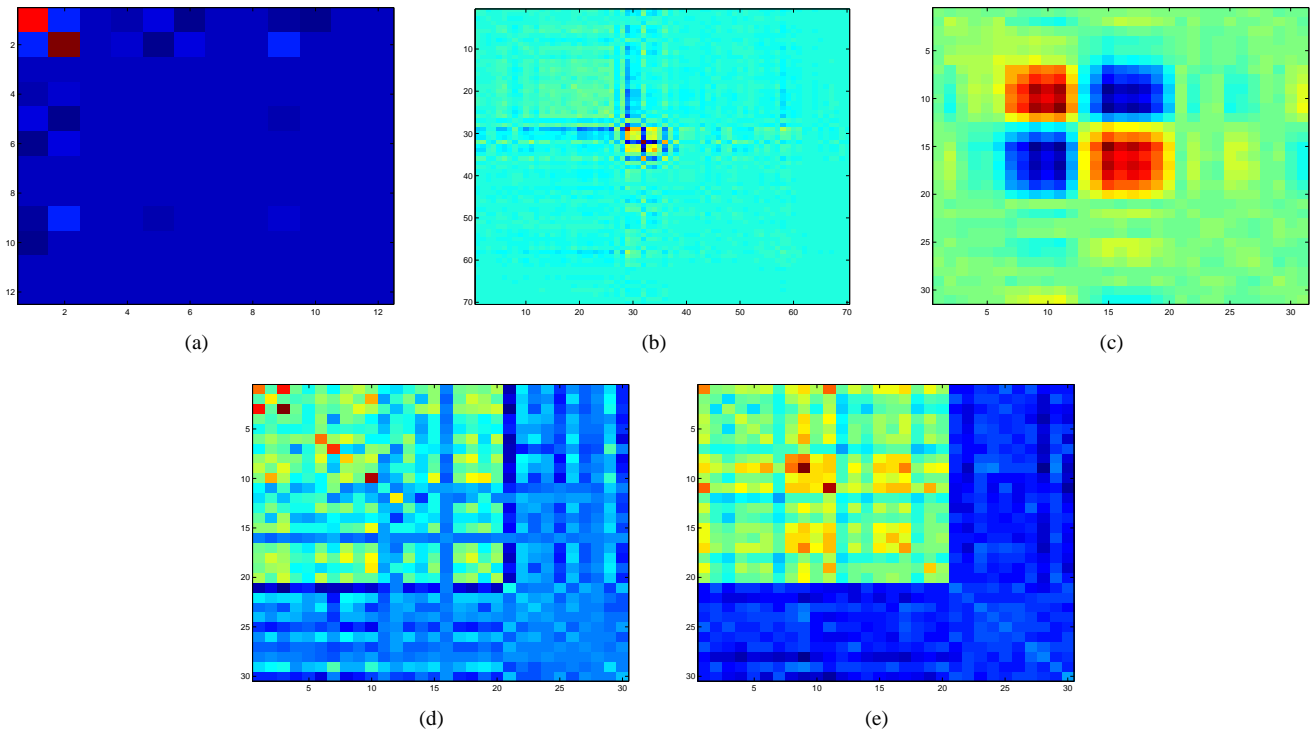


Figure 3: Distance metric matrices learned on (a) *banana*, (b) *splice*, (c) *waveform*, (d) *ringnorm*, and (e) *twonorm*. The lower right corner (the last 10 rows and columns) corresponds to irrelevant features, and thus takes small values near zero.

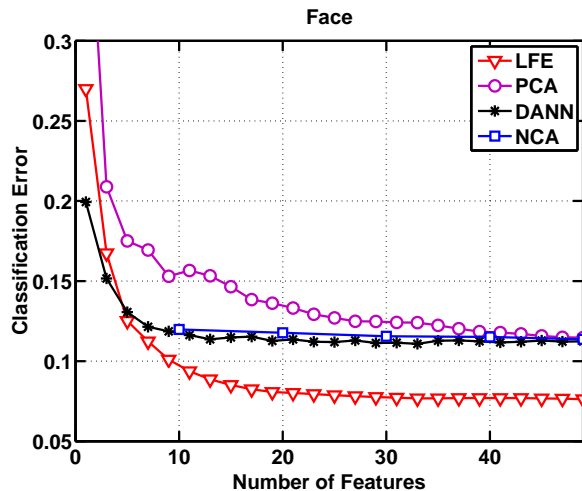


Figure 4: Classification errors on the *face* dataset.

algorithm that uses a distance function to define similarities among patterns can benefit from distance metric learning. For example, in SVM with RBF kernel, the classification performance relies on the accurate estimation of a kernel matrix, the ij -th entry of which is computed as $K_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\delta)$ where δ is the kernel width. It is difficult to directly learn a distance metric in the SVM framework since the margin maximized by SVM is nonlinearly related to a distance metric through a kernel matrix. Below, we conduct an experiment on using LFE to improve the performance of SVM. Due to the high computational cost involved in the estimation of the structural parameters of SVM (i.e. the kernel width and the regularization parameter) that are determined by grid searching through 10-fold cross-validation on the training data for each feature extraction algorithms, we only perform SVM on two levels of feature numbers. With 20 features, the testing errors (STDs) of LFE, DANN and PCA are 5.9 (0.9)%, 6.6 (1.0)% and 7.5 (0.9)%, respectively, and with 50 features, 4.1 (0.8)%, 4.5 (0.8)% and 4.9 (0.8)%, respectively. Though the performance difference between PCA and LFE is diminished, largely due to the robustness of SVM, LFE performs significantly better than PCA (p-value < 0.01).

6 Conclusion

In this paper, we have provided a very easy-to-understand interpretation for RELIEF that explains its success in practical applications. The mathematical interpretation has motivated us to propose a new feature extraction algorithm as a natural generalization of RELIEF. Compared to PCA, our algorithm also has a clear physical meaning and can be implemented easily with a comparable computational cost. Unlike many other feature selection algorithms, by approximately optimizing the leave-one-out accuracy of the nearest neighbor classifier, our algorithm provides a built-in mechanism for automatically removing irrelevant features during learning. We have experimentally demonstrated that LFE performs significantly better than PCA, NCA and DANN. Given the popularity of PCA and the excellent performance of LFE, we believe that our algorithm should find widespread use in similar applications.

bor classifier, our algorithm provides a built-in mechanism for automatically removing irrelevant features during learning. We have experimentally demonstrated that LFE performs significantly better than PCA, NCA and DANN. Given the popularity of PCA and the excellent performance of LFE, we believe that our algorithm should find widespread use in similar applications.

References

- [1] R. Kohavi and G. H. John, *Wrappers for feature subset selection*, *Artifi. Intell.*, 97 (1997), pp. 273–324.
- [2] P. Pudil, J. Novovicova, and J. Kittler, *Floating search methods in feature selection*, *Pattern Recog. Lett.*, 15 (1994), pp. 1119–1125.
- [3] K. Kira and L. A. Rendell, *A practical approach to feature selection*, *Proc. 9th Int. Conf. Mach. Learn.*, (1992), pp. 249–256.
- [4] T. G. Dietterich, *Machine learning research: Four current directions*, *AI Mag.*, 18 (1997), pp. 97–136.
- [5] Y. Sun and J. Li, *Iterative RELIEF for feature weighting*, *Proc. 21st Int. Conf. Mach. Learn.*, (2006), pp. 913–920.
- [6] D. Wettschereck, D. W. Aha, T. Mohri, (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms, *Artif. Intell. Rev.*, 11 (1997), pp. 273–314.
- [7] R. Gilad-Bachrach, A. Navot, and N. Tishby, *Margin based feature selection-theory and algorithms*, *Proc. 19th Int. Conf. Mach. Learn.*, (2004), pp. 43–50.
- [8] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization*, John Wiley and Sons Inc, New York, 2001.
- [9] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, New York, 2003.
- [10] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, *Neighbourhood components analysis*, *Advances in Neural Information Processing Systems*, (2005), pp. 513–520.
- [11] K. Q. Weinberger, J. Blitzer, and L. K. Saul, *Distance metric learning for large margin nearest neighbor classification*, *Advances in Neural Information Processing Systems*, (2006).
- [12] C. Blake and C. Merz, *UCI repository of machine learning databases*, 1998.
- [13] A. Martinez and R. Benavente, *The AR-face database*, tech. report 24, 1998.
- [14] I. Kononenko, *Estimating attributes: Analysis and extensions of RELIEF*, *Proc. Euro. Conf. Mach. Learn.*, (1994), pp. 171–182.