## Feature Extraction through Local Learning<sup>\*</sup>

Yijun  $Sun^{\dagger}$ , Dapeng  $Wu^{\ddagger}$ 

<sup>†</sup>Interdisciplinary Center for Biotechnology Research <sup>‡</sup>Department of Electrical and Computer Engineering University of Florida Gainesville, FL 32610-3622

Abstract: RELIEF is considered one of the most successful algorithms for assessing the quality of features. It has been recently proven that RELIEF is an online learning 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 can be solved as an eigenvalue decomposition problem with a closed-form solution. A fast implementation of LFE is derived. Compared to PCA, LFE also has a clear physical meaning and can be implemented easily with a comparable computational cost. Compared to other feature extraction algorithms, LFE has an explicit mechanism to remove irrelevant features. Experiments on synthetic and real-world data are presented. The results demonstrate the effectiveness of the proposed algorithm.

#### Statistical Analysis and Data Mining

Submitted in May 2008, revised in September 2008, accepted in January 2009

<sup>\*</sup>This work was supported in part by the Komen Breast Cancer Foundation under grant No. BCTR0707587. Please address all correspondence to: Dr. Yijun Sun, Interdisciplinary Center for Biotechnology Research, University of Florida, P.O. Box 103622, Gainesville, FL 32610-3622, USA. E-mail: sun@dsp.ufl.edu.

## 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.

Suppose we are given a training dataset  $\mathcal{D}=\{(\mathbf{x}_n, y_n)\}_{n=1}^N \in \mathcal{X} \times \mathcal{Y}$ , where  $\mathcal{X} \in \mathcal{R}^I$  is the pattern space, I is the feature dimensionality, and  $\mathcal{Y}$  is the label space. 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

$$\mathcal{L}(\mathbf{P}) = \sum_{n=1}^{N} \ell(\mathbf{P}^T \mathbf{x}_n, y_n) , \qquad (1)$$

where  $\ell(\cdot)$  is a criterion function, and **P** is a projection matrix. By convention, pattern vector  $\mathbf{x}_n$  is denoted as a column. 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, a classification algorithm is employed to evaluate the goodness of a selected feature subset, whereas in filter methods criterion functions evaluate feature subsets by their information content, typically interclass distance (e.g., Fisher score) or statistical measures (e.g., p-value of t-test), instead of optimizing the performance of any specific learning algorithm directly. Hence, filter methods are computationally much more efficient, but usually do not perform as well as wrapper methods. In addition to a criterion function, a search strategy is also needed. An exhaustive search is optimum but quickly becomes computationally infeasible with the increase of problem size. Therefore, 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. For more detailed discussions, interested readers can refer to [3, 4] and the references therein.

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 limited set {0, 1}. This strategy has many advantages. For example, there is no need to pre-specify the number of relevant features. Also, standard optimization techniques (e.g., gradient descent) can be used to avoid combinatorial search. Among the existing feature weighting algorithms, the RELIEF algorithm [5] is considered one of the most successful ones due to its simplicity and effectiveness [6]. RELIEF has been long regarded as a heuristic filter method until recently we have mathematically proved that RELIEF is an online learning algorithm that solves a convex optimization problem aimed at maximizing a margin-based objective function [7]. The margin is defined based on 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 solving a convex problem, RELIEF avoids any exhaustive or heuristic combinatorial search, and thus can be implemented efficiently. These two merits clearly explain the success of RELIEF in practical applications.

One major shortcoming of feature weighting and selection, however, is their inability to capture the interaction of correlated features [8]. In some applications, such as face recognition, where to preserve the 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 projection matrix. LFE collects discriminant information through local learning and can be solved as an eigenvalue decomposition problem 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 (PCA). As with RELIEF, LFE learns a distance metric matrix to approximately optimize the leave-one-out accuracy of a nearest neighbor classifier, and thus has an explicit mechanism to remove irrelevant features. Experiments on synthetic and real-world data are presented. The results demonstrate the effectiveness of the proposed algorithm.

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 is discussed. In Section 4, we extend the algorithm to address multiclass problems. In Section 5, some related work are briefly reviewed. In Section 6, numerical experiments are performed using both synthetic and real-world data, and finally the paper is concluded in Section 7.

### 2 Optimization Approach to RELIEF

We first present a brief review of RELIEF. At the moment, we only consider binary problems, while multiclass problems are addressed in Section 4. The pseudo-code of RELIEF is presented in Fig. 1. 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 different class (termed the *nearest miss* or NM). The weight of the *i*-th feature is then updated as:  $w_i = w_i + |\mathbf{x}^{(i)} - \text{NM}^{(i)}(\mathbf{x})| - |\mathbf{x}^{(i)} - \text{NH}^{(i)}(\mathbf{x})|$ , where  $\mathbf{x}^{(i)}$  is the *i*-th component of pattern  $\mathbf{x}$ . After a feature weight vector is learned, the features are ranked based on their corresponding weight values, and usually only the features with positive weights are used for classification.

We below present a mathematical interpretation for the seemingly heuristic RELIEF algorithm to show why it works. Following the margin definition in [9], we define the margin for pattern  $\mathbf{x}_n$  as

$$\rho_n = d(\mathbf{x}_n - \mathrm{NM}(\mathbf{x}_n)) - d(\mathbf{x}_n - \mathrm{NH}(\mathbf{x}_n)) , \qquad (2)$$

where  $NM(\mathbf{x}_n)$  and  $NH(\mathbf{x}_n)$  are the nearest miss and hit of pattern  $\mathbf{x}_n$ , respectively, and  $d(\cdot)$  is a distance function. For the purpose of this paper, we define  $d(\mathbf{x}) = \sum_i |x_i|$ ,

Algorithm 1: RELIEF Algorithm **Input** : Data  $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ , number of iterations T**Output**: Feature weights **w** 1 Initialization: Set  $\mathbf{w}^{(0)} = \mathbf{0}, t = 1$ ; 2 repeat Randomly select a pattern  $\mathbf{x}$  from  $\mathcal{D}$ ; 3 Find the nearest hit  $NH(\mathbf{x})$  and miss  $NM(\mathbf{x})$  of  $\mathbf{x}$ ; 4 for i = 1 : I do  $\mathbf{5}$  $w_i = w_i + |\mathbf{x}^{(i)} - \mathrm{NM}^{(i)}(\mathbf{x})| - |\mathbf{x}^{(i)} - \mathrm{NH}^{(i)}(\mathbf{x})|;$ 6 end 7 t = t + 1;8 9 until t > T;

#### Figure 1: Pseudo-code of RELIEF

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 using the training data *excluding* pattern  $\mathbf{x}_n$ . One natural idea then is to scale each feature, and thus obtain a weighted feature space, parameterized by a nonnegative vector  $\mathbf{w}$ , so that the *leave-one-out* error  $\sum_{n=1}^{N} I(\rho_n(\mathbf{w}) < 0)$  in the induced feature space is minimized, where  $I(\cdot)$  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:

$$\max_{\mathbf{w}} \sum_{n=1}^{N} \rho_n(\mathbf{w}) = \sum_{n=1}^{N} \left( \sum_{i=1}^{I} w_i |\mathbf{x}_n^{(i)} - \mathrm{NM}^{(i)}(\mathbf{x}_n)| - \sum_{i=1}^{I} w_i |\mathbf{x}_n^{(i)} - \mathrm{NH}^{(i)}(\mathbf{x}_n)| \right)$$
(3)  
subject to  $\|\mathbf{w}\|_2^2 = 1, \mathbf{w} \ge 0$ ,

where the constraint  $\|\mathbf{w}\|_2^2 = 1$  prevents the maximization from increasing without bound, and  $\mathbf{w} \ge 0$  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. (3) can be simplified to read

$$\max_{\mathbf{w}} \quad \mathbf{w}^{\mathrm{T}} \mathbf{z}$$
subject to  $\|\mathbf{w}\|_{2}^{2} = 1, \mathbf{w} \ge 0$ . (4)

The Lagrangian of Eq. (4) is

$$L = -\mathbf{w}^{\mathrm{T}}\mathbf{z} + \lambda(\|\mathbf{w}\|_{2}^{2} - 1) + \sum_{i=1}^{I} \theta_{i}(-w_{i}) , \qquad (5)$$

where  $\lambda$  and  $\theta \ge 0$  are Lagrangian multipliers. Taking the derivative of L with respect to **w** and setting it to zero yield

$$\frac{\partial L}{\partial \mathbf{w}} = -\mathbf{z} + 2\lambda \mathbf{w} - \boldsymbol{\theta} = 0 \quad \Rightarrow \quad \mathbf{w} = \frac{1}{2\lambda} (\mathbf{z} + \boldsymbol{\theta}) .$$
 (6)

Below, we derive a closed-form solution for  $\mathbf{w}$ . We first make an assumption that there exists  $z_i > 0$ . This assumption is very weak since if it does not hold (i.e.,  $\mathbf{z} \leq 0$ ), it simply says that on average the distance between a pattern and its nearest miss is smaller than the distance of the pattern from its nearest hit, which is very rare in real applications. In this case, machine learning algorithms that make decisions based on neighborhood information (e.g., KNN and SVM with RBF kernel) will perform poorly. With the above assumption, we prove that  $\lambda > 0$  by contradiction. Suppose  $\lambda < 0$ . Since there exists  $z_i > 0$ , then  $z_i + \theta_i > 0$  and  $w_i = (z_i + \theta_i)/2\lambda < 0$ , which contradicts the constraint  $\mathbf{w} \geq 0$ . By using the Karush-Kuhn-Tucker (KKT) condition [10], namely  $\sum_i \theta_i w_i = 0$ , it is easy to verify the following three cases:

- Case 1:  $z_i = 0 \Rightarrow \theta_i = 0, w_i = 0;$
- Case 2:  $z_i > 0 \Rightarrow z_i + \theta_i > 0 \Rightarrow w_i > 0 \Rightarrow \theta_i = 0;$

1

• Case 3:  $z_i < 0 \Rightarrow \theta_i > 0 \Rightarrow w_i = 0 \Rightarrow z_i = -\theta_i$ .

It immediately follows that the optimum solution of  $\mathbf{w}$  can be calculated in the following closed form:

$$w_i = \begin{cases} 0 & \text{if } z_i \le 0\\ \frac{1}{2\lambda} z_i & \text{if } z_i > 0 \end{cases}$$

and

$$\mathbf{w}^* = \mathbf{w} / \|\mathbf{w}\|_2 = (\mathbf{z})^+ / \|(\mathbf{z})^+\|_2 , \qquad (7)$$

where  $(z_i)^+ = \max(z_i, 0)$ . By comparing the expression of  $\mathbf{w}^*$  with the weight update rule of RELIEF (line 6 in Fig. 1), we conclude that RELIEF is an online learning algorithm that solves the optimization scheme of Eq. (3). This is true except when  $w_i^* = 0$  for  $z_i < 0$ , which usually corresponds to irrelevant features.

RELIEF maximizing the averaged margin was first observed in [9], but no mathematical proof was provided. In the literature, RELIEF is usually regarded as a filter method. From our analysis, we find that RELIEF is a feature weighting algorithm that utilizes the performance of a nonlinear classifier when 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 now consider the following optimization problem:

$$\max_{\mathbf{W}} \sum_{n=1}^{N} \rho_n(\mathbf{W}) = \sum_{n=1}^{N} \mathbf{m}_n^T \mathbf{W} \mathbf{m}_n - \sum_{n=1}^{N} \mathbf{h}_n^T \mathbf{W} \mathbf{h}_n$$
subject to  $\|\mathbf{W}\|_F^2 = 1, \mathbf{W} \ge 0$ ,
(8)

where  $\rho_n(\mathbf{W})$  is the margin of pattern  $\mathbf{x}_n$  computed with respect to  $\mathbf{W}$ ,  $\mathbf{m}_n = \mathbf{x}_n - \text{NM}(\mathbf{x}_n)$ ,  $\mathbf{h}_n = \mathbf{x}_n - \text{NH}(\mathbf{x}_n)$ , 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 eigenvalues of  $\mathbf{W}$ . The optimization problem in Eq. (8) has the same physical meaning as that in Eq. (3).

The direct optimization of Eq. (8) is computationally arduous. We therefore derive a closed-form solution. Some organization is merited. First, since **W** is required to be a distance metric matrix, it is symmetric positive and semidefinite, and thus can be written as  $\mathbf{W} = \mathbf{P}\mathbf{P}^T$ , where  $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_I]$ . We further assume that  $\{\mathbf{p}_i\}_{i=1}^I$  are pairwise

orthogonal, that is,  $\langle \mathbf{p}_i, \mathbf{p}_j \rangle = 0$ , for any  $i \neq j$ . This can be easily done through the eigenvalue decomposition of **W** as follows:

$$\mathbf{W} = \boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^{T} = \boldsymbol{\Phi} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{\Phi} ,$$
  
$$= [\sqrt{\lambda_{1}} \boldsymbol{\phi}_{1}, \cdots, \sqrt{\lambda_{I}} \boldsymbol{\phi}_{I}] [\sqrt{\lambda_{1}} \boldsymbol{\phi}_{1}, \cdots, \sqrt{\lambda_{I}} \boldsymbol{\phi}_{I}]^{T} ,$$
(9)

Here,  $\{\phi_i\}_{i=1}^I$  and  $\{\lambda_i\}_{i=1}^I$  are the eigenvectors and eigenvalues of **W**, respectively. Let

$$\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_I] = [\sqrt{\lambda_1} \phi_1, \cdots, \sqrt{\lambda_I} \phi_I] .$$
(10)

The task then is to solve for the set  $\{\mathbf{p}_i\}_{i=1}^I$ . The term  $\sum_{n=1}^N \mathbf{m}_n^T \mathbf{W} \mathbf{m}_n$  in Eq. (8) can be rearranged as

$$\sum_{n=1}^{N} \mathbf{m}_{n}^{T} \mathbf{W} \mathbf{m}_{n}$$

$$= \operatorname{tr} \left( \mathbf{W} \sum_{n=1}^{N} \mathbf{m}_{n} \mathbf{m}_{n}^{T} \right),$$

$$= \operatorname{tr} \left( \mathbf{W} \boldsymbol{\Sigma}_{m} \right), \qquad \left( \boldsymbol{\Sigma}_{m} \triangleq \sum_{n=1}^{N} \mathbf{m}_{n} \mathbf{m}_{n}^{T} \right)$$

$$= \operatorname{tr} \left( \boldsymbol{\Sigma}_{m} \sum_{i=1}^{I} \mathbf{p}_{i} \mathbf{p}_{i}^{T} \right),$$

$$= \sum_{i=1}^{I} \mathbf{p}_{i}^{T} \boldsymbol{\Sigma}_{m} \mathbf{p}_{i}.$$
(11)

Hence, the objective function of Eq. (8) can be expressed as follows:

$$\sum_{n=1}^{N} \mathbf{m}_{n}^{T} \mathbf{W} \mathbf{m}_{n} - \sum_{n=1}^{N} \mathbf{h}_{n}^{T} \mathbf{W} \mathbf{h}_{n} ,$$

$$= \sum_{i=1}^{I} \mathbf{p}_{i}^{T} \left( \mathbf{\Sigma}_{m} - \mathbf{\Sigma}_{h} \right) \mathbf{p}_{i} . \qquad \left( \mathbf{\Sigma}_{h} \triangleq \sum_{n=1}^{N} \mathbf{h}_{n} \mathbf{h}_{n}^{T} \right)$$
(12)

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

$$\|\mathbf{W}\|_{F}^{2} = \sum_{s,t} w_{s,t}^{2} = \sum_{s,t} \left(\sum_{i} \mathbf{p}_{i}^{(s)} \mathbf{p}_{i}^{(t)}\right)^{2} ,$$
  
$$= \sum_{i,j} \left(\sum_{s} \mathbf{p}_{i}^{(s)} \mathbf{p}_{j}^{(s)}\right)^{2} = \sum_{i,j} \left(\mathbf{p}_{i}^{T} \mathbf{p}_{j}\right)^{2} ,$$
(13)

where  $\mathbf{p}_{i}^{(s)}$  is the *s*-th component of vector  $\mathbf{p}_{i}$ . An equivalent optimization problem of Eq. (8) can thus be formulated as

$$\max_{\{\mathbf{p}_i\}_{i=1}^{I}} \sum_{i=1}^{I} \mathbf{p}_i^T \left( \boldsymbol{\Sigma}_m - \boldsymbol{\Sigma}_h \right) \mathbf{p}_i ,$$
  
subject to 
$$\sum_{i,j} \left( \mathbf{p}_i^T \mathbf{p}_j \right)^2 = \sum_i \left( \mathbf{p}_i^T \mathbf{p}_i \right)^2 = 1 ,$$
 (14)

where the equality in the constraint is due to the orthogonality of  $\{\mathbf{p}_i\}_{i=1}^I$ . 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. (14) is:

$$L = -\sum_{i=1}^{I} \mathbf{p}_{i}^{T} \left( \boldsymbol{\Sigma}_{m} - \boldsymbol{\Sigma}_{h} \right) \mathbf{p}_{i} + \lambda \left( \sum_{i=1}^{I} \left( \mathbf{p}_{i}^{T} \mathbf{p}_{i} \right)^{2} - 1 \right) \,. \tag{15}$$

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

$$\partial L/\partial \mathbf{p}_{i} = -2\Sigma_{mh}\mathbf{p}_{i} + 4\lambda \mathbf{p}_{i}^{T}\mathbf{p}_{i}\mathbf{p}_{i} = 0,$$
  

$$\Rightarrow \Sigma_{mh}\mathbf{p}_{i}/\|\mathbf{p}_{i}\|_{2} = 2\lambda \|\mathbf{p}_{i}\|_{2}^{2}\mathbf{p}_{i}/\|\mathbf{p}_{i}\|_{2},$$
  

$$\Rightarrow \Sigma_{mh}\bar{\mathbf{p}}_{i} = 2\lambda \|\mathbf{p}_{i}\|_{2}^{2}\bar{\mathbf{p}}_{i}, \text{ (Assume } \|\mathbf{p}_{i}\|_{2} \neq 0.)$$
(16)

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

$$\mathbf{p}_i = \sqrt{\beta_i} \bar{\mathbf{p}}_i, \ \beta_i \ge 0, \ 1 \le i \le I \ . \tag{17}$$

Substituting Eq. (17) into Eq. (14), we obtain an optimization problem of the following form:

$$\max_{\boldsymbol{\beta}} \qquad \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\sigma} ,$$
subject to  $\|\boldsymbol{\beta}\|_{2}^{2} = 1, \boldsymbol{\beta} \ge 0 .$ 

$$(18)$$

This problem is identical to the one expressed by Eq. (4). Hence, the conclusion drawn in the last section can be directly applied here:

For  $\{i | \sigma_i > 0, 1 \le i \le I\}$ :

$$\beta_i = \sigma_i / \sqrt{\sum_{\{j \mid \sigma_j > 0\}} \sigma_j^2} \tag{19}$$

$$\mathbf{p}_{i} = \bar{\mathbf{p}}_{i} \sqrt{\sigma_{i} / \sqrt{\sum_{\{j \mid \sigma_{j} > 0\}} \sigma_{j}^{2}}}$$
(20)

For  $\{i | \sigma_i \leq 0, 1 \leq i \leq I\}$ :

$$\beta_i = 0 \quad \text{and} \quad \mathbf{p}_i = \mathbf{0} \;.$$
 (21)

Finally, note that the orthogonality constraint of  $\{\mathbf{p}_i\}_{i=1}^I$  is automatically fulfilled.

#### 3.1 Feature Extraction

One direct application of Eq. (8) is for feature extraction. The physical meaning of Eq. (8) is that we project the data onto a subspace spanned by  $\{\mathbf{p}_i\}$  so that the average margin  $\frac{1}{N} \sum_{n=1}^{N} \rho_n(\mathbf{W})$  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. (8) as the discriminant power, which can be shown to be  $\sqrt{\sum_{\{i|\sigma_i>0\}} \sigma_i^2}$  by plugging Eqs. (19), (20) and (21) into Eq. (18). Forming a projection matrix  $\mathbf{P}$  by only using  $\{\mathbf{p}_i\}$ corresponding to the largest eigenvalues of  $\Sigma_{mh}$ , the suffered discriminant power loss is proportional to the sum of the squared positive eigenvalues that are discarded. More precisely, let  $\sigma_1 \geq \cdots \geq \sigma_T > 0 > \sigma_{T+1} \geq \cdots \geq \sigma_I$ . By using the first t eigenvectors with t < T, the discriminant power loss is:

$$\sum_{i=t+1}^{T} \sigma_i^2 / \sqrt{\sum_{\{j|\sigma_j>0\}} \sigma_j^2} .$$

$$(22)$$

We name the newly proposed algorithm as Local Feature Extraction, or LFE for short, since the discriminant information is collected through local learning. LFE may be viewed as the counterpart of LLE [11] in the context of supervised learning. LLE is a manifold learning algorithm used to reduce data dimensionality in unsupervised learning. Though both algorithms differ completely in their motivations and detailed algorithm derivations, it is interesting to note that both LLE and LFE perform learning locally and end up with an eigenvalue decomposition problem.

LFE has several nice properties. First, since local learning is a nonlinear process, LFE is capable of extracting nonlinear discriminant information. It examines the nearest miss and hit of each training sample, and pools the local discriminant information over all points of the training data into matrix  $\Sigma_{mh}$  to determine a projection matrix, a procedure similar to that of the DANN algorithm proposed by [12]. This property is empirically

verified in Fig. 3. 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, as with PCA, the discriminant power of the features extracted by LFE can be ranked based on the magnitude of the corresponding eigen-values. This property offers some computational savings in the estimation of the optimum number of features: one first learns a full distance metric matrix W, and then sequentially includes the projection vectors  $\{\mathbf{p}_i\}$  based on their discriminant power (i.e., eigen-value  $\sigma_i$ ) into a projection matrix and evaluates its classification performance by using a cross-validation method until the optimum number of features is found. Fourth, LFE has an *explicit* mechanism to eliminate irrelevant features, a nice property inherited from RELIEF. It can be explained as follows: if  $\mathbf{W}$  in Eq. (8) was constrained to be a diagonal matrix, then the solution of Eq. (8) is the same as that of Eq. (3). Hence, one would expect that, when the diagonal constraint is lifted, the diagonal elements of W are approximately equal to the weights learned by RELIEF. This is empirically demonstrated in Fig. 5. This means that LFE is capable of removing irrelevant features by assigning them small values near zero. Also note that some  $\beta_i$ 's in Eqs. (19) and (21) may take the value of zero, which means that LFE is an incomplete linear transformation and can automatically null out the noise subspaces that contain little discriminant information. This property is experimentally verified in our experiment (see Fig. 5).

#### **3.2** Fast Implementation of LFE

The major computation complexity of LFE comes from the eigenvalue decomposition of  $\Sigma_{mh}$ . Therefore, LFE has the same computational complexity as PCA. However, in many practical applications, such as face recognition (c.f. Table 1), 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 we herein only present a fast implementation for LFE.

We first denote  $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ . For notational simplicity, we also assume that the data has been centered, that is,  $\sum_{n=1}^{N} \mathbf{x}_n = \mathbf{0}$ . Let  $\mathbf{X}\mathbf{X}^T = \mathbf{\Psi}\mathbf{\Delta}\mathbf{\Psi}^T$  be the eigenvalue decomposition of the scatter matrix  $\mathbf{X}\mathbf{X}^T$ , where the eigenvalues in  $\mathbf{\Delta}$  are sorted in a decreasing order. The pattern  $\mathbf{x}$  can then be expressed as  $\mathbf{x} = \mathbf{\Psi}_1 \mathbf{y}$ , where  $\mathbf{\Psi}_1$  contains the eigenvectors corresponding to the first N largest eigenvalues of  $\mathbf{X}\mathbf{X}^T$  and  $\mathbf{y} = \mathbf{\Psi}_1^T \mathbf{x}$ . Then, Eq. (16) can be expressed as

$$\Psi \begin{bmatrix} \Sigma_{mhy} & \mathbf{0} \\ \mathbf{0}^T & \mathbf{0} \end{bmatrix} \Psi^T \bar{\mathbf{p}}_i = \sigma_i \bar{\mathbf{p}}_i , \qquad (23)$$

where  $\Sigma_{mhy}$  is the same as  $\Sigma_{mh}$ , but is computed with **y**. Defining  $\tilde{\mathbf{p}}_i = \Psi^T \bar{\mathbf{p}}_i$  and premultiplying  $\Psi^T$  to both sides of Eq. (23) yield

$$\begin{bmatrix} \boldsymbol{\Sigma}_{mhy} & \mathbf{0} \\ \mathbf{0}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{p}}_{i,1} \\ \tilde{\mathbf{p}}_{i,2} \end{bmatrix} = \sigma_i \begin{bmatrix} \tilde{\mathbf{p}}_{i,1} \\ \tilde{\mathbf{p}}_{i,2} \end{bmatrix} .$$
(24)

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

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 Extension to Address Multiclass Problems

The newly proposed LFE algorithm can be easily extended to address multiclass problems. A natural extension of the margin defined in Eq. (2) to multiclass problems is [13]:

$$\rho_{n}(\mathbf{w}) = \min_{\{c \in \mathcal{Y}, c \neq y_{n}\}} d(\mathbf{x}_{n}, \mathrm{NM}^{(c)}(\mathbf{x}_{n}) | \mathbf{w}) - d(\mathbf{x}_{n}, \mathrm{NH}(\mathbf{x}_{n}) | \mathbf{w}) ,$$
  
$$= \min_{\mathbf{x}_{i} \in \mathcal{D} \setminus \mathcal{D}_{y_{n}}} d(\mathbf{x}_{n}, \mathbf{x}_{i} | \mathbf{w}) - d(\mathbf{x}_{n}, \mathrm{NH}_{n} | \mathbf{w}) ,$$
(25)

where  $\mathcal{Y}$  is the set of class labels,  $\mathrm{NM}^{(c)}(\mathbf{x}_n)$  is the nearest neighbor of  $\mathbf{x}_n$  from class c, and  $\mathcal{D}_c$  is a subset of  $\mathcal{D}$  containing only samples from class c. The derivation of our feature extraction algorithm for multiclass problems by using the margin defined in Eq. (25) is straightforward.

## 5 Related Work

In this section, we briefly review some feature extraction algorithms that we will compare to LFE in our experiments. PCA is probably one of the most commonly used algorithms in the literature for feature extraction, 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 [14]) 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 thus suffer from the same problems as PCA. Linear discriminant analysis (LDA) tries to overcome this problem of PCA by projecting data onto a subspace such 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 fails when the feature dimensionality is larger than the sample size. To overcome this limitation, a new algorithm, referred to as MMC, is recently proposed in [15]. MMC maximizes the trace of the between-class scatter matrix minus the within-class scatter matrix. It avoids the inversion of the within-class scatter matrix, and thus works well in problems with a small sample size. However, as with LDA, MMC can at most generate C - 1 features, which do not necessarily encode all of the discriminant information, where C is the number of classes. Moreover, when the number of samples exceeds the data dimensionality, MMC is equivalent to LDA |15|.

In [12], a local learning based feature extraction algorithm, referred to as DANN, is proposed. DANN projects data onto a subspace corresponding to the largest eigenvectors of the average local between-class scatter matrices. Therefore, DANN is highly related to LDA. Favorable results have been reported in [12, 16], indicating that DANN has the capability to extract discriminant information while filtering out irrelevant features. However, the objective function optimized by DANN is not directly related to the performance of any specified 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 estimate of the number of extracted features.

More recently, a distance metric learning algorithm named neighborhood component analysis (NCA) is proposed in [17]. The basic idea of NCA is to estimate a distance 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. Therefore, NCA does not scale well to a classification problem with a large feature dimensionality. For example, in a face recognition problem that we consider in our experiment (c.f. Table 1), the images are of size  $85 \times 60$ . To learn a full metric matrix, NCA essentially performs gradient-descent search in a space of size  $5100 \times 5100$ , which is computationally infeasible.

Another related algorithm is LMNN [18]. 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 [18].

Two local-learning based classification algorithms, namely ADAMENN [19] and LAMANNA [20], are worthy of mention. Similar to LFE, both algorithms have an embedded adaptive mechanism that learns discriminant information locally. However, both algorithms perform feature weighting, instead of feature extraction, which is the main focus of the paper. Moreover, the way that both ADAMENN and LAMANNA use to determine feature relevance is very different from LFE. For example, LAMANNA first trains a SVM to construct a deci-

data sets	train	test	features	classes
banana	400	4900	2	2
twonorm	400	7000	20	2
waveform	400	4600	21	2
ringnorm	400	7000	20	2
splice	1000	2175	60	2
thyroid	140	75	5	2
USPS (3 vs 5)	1214	326	256	2
face	500	1456	5100	2
11 Tumors	174	/	12533	11
9 Tumors	60	/	5726	9
Brain Tumors 1	90	/	5920	5
Brain Tumors 2	90	/	10367	4
Leukemia 1	72	/	5327	3
Leukemia 2	72	/	11225	3
Lung Cancer	203	/	12600	5
DLBCL	77	/	5469	2

Table 1: Data Summary

sion boundary, then estimates a feature weighting vector for each test sample by computing the gradient vector of the closest point of the test sample on the decision boundary, and finally the feature weighting vector is used to classify the test sample by using a K-nearest neighbor classifier. Another major difference is that the two algorithms are designed for classification purposes while LFE is a data preprocessing method. For this reason, we do not perform a comparison between LFE and the two algorithms in our experiments.

# 

Figure 2: *USPS* handwritten digits: "3" (top row) and "5" (bottom row), and *face*: female (top row) and male (bottom row).

## 6 Experiments

In this section, we conduct some experiments to demonstrate the effectiveness of LFE using a wide variety of synthetic and real-world data.

#### 6.1 Experiments on UCI Datasets

We first compare LFE with RELIEF, PCA, NCA, MMC, and DANN on six UCI data sets [21], including banana, twonorm, waveform, ringnorm, thyroid, and splice. The data information is summarized in Table 1. 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. The code of NCA is downloaded from [17], and the default settings are used throughout the study. We compare RELIEF and LFE to illustrate when and how LFE can improve classification performances of RELIEF. It should be emphasized that feature weighting and extraction are used in different scenarios. In the experiment, we actually use RELIEF-F [22], which uses M, instead of just one, nearest hits and misses in computing margins to ensure greater robustness of the algorithm against noise. For the same reason, we search for multiple nearest neighbors in LFE. The value of M is found through 10-fold cross-validation on training data.

KNN is used to estimate the classification errors for 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 machine learning algorithms. For this reason, many distance metric learning algorithms (e.g., [17,

Table 2: The testing errors and STDs (%) on six UCI datasets. When the difference between weighted and unweighed approaches (KNN performed on noisy 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-valu	ie level.						
Data	KNN	KNN	RELIEF	MMC	DANN	NCA	LFE	p-value	p-value
	(clean data)	(noisy data)						(LFE/RELIEF)	(LFE/NCA)
banana	11.6(0.6)	37.1(2.1)	12.3(0.7)	47.7(1.0)	23.0(4.4)	25.4(4.4)	18.2(2.5)	$< 0.001^{-1}$	$< 0.001^{+}$
splice	26.7(2.0)	26.6(1.8)	11.4(0.9)	16.9(4.1)	14.1(0.7)	13.9(0.4)	12.0(0.7)	$0.02^{-}$	$0.002^{+}$
wave form	11.4(0.5)	12.6(0.7)	10.8(0.5)	17.4(3.6)	11.1(0.8)	10.9(0.5)	9.8(0.6)	$< 0.001^{+}$	$< 0.001^{+}$
ringnorm	34.7(1.7)	39.2(1.3)	31.3(2.8)	33.8(2.1)	24.1(2.7)	26.8(1.2)	22.0(1.3)	$< 0.001^{+}$	$< 0.001^{+}$
twonorm	3.3(0.2)	3.9(0.3)	3.5(0.4)	2.6(0.3)	2.9(0.3)	3.8(0.5)	2.6(0.2)	$< 0.001^{+}$	$< 0.001^{+}$
thy roid	4.4(2.4)	17.0(3.0)	9.5(3.0)	13.7(8.5)	8.8(3.4)	7.0(2.6)	6.2(2.8)	$0.001^{+}$	$0.36^{+}$

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

Dataset	NCA	DANN	LFE	PCA	MMC
banana	12.6	3.6	3.3	3.0	3.1
twonorm	41.6	4.7	4.8	4.7	4.7
waveform	38.8	4.4	4.3	4.2	4.2
ringnorm	39.4	4.7	4.7	4.7	4.7
splice	383.6	8.4	8.4	4.8	4.9
thyroid	4.9	3.2	3.3	3.2	3.2
USPS	1120	22.3	21.5	4.2	4.6
face	185.7	24.9	23.8	8.6	8.7

18]) 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 6.2, 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



Figure 3: The original *banana* data and the top two features extracted by PCA, DANN, NCA, and LFE using corrupted *banana* data. The top two features extracted by LFE faithfully reconstruct the original data except for some minor distortions and rotation.



Figure 4: Classification errors of PCA, LFE, NCA, MMC and DANN on six UCI datasets.

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. 4. MMC can extract only one feature since all of the UCI data are binary problems. In Table 2, the testing errors and the standard deviations (STDs) are reported. For RELIEF, after finding feature weights, we first remove the features corresponding to the negative weights, and then perform classification on the weighted feature space [8]. For both DANN and LFE, the optimum number of features used in KNN is estimated through cross-validation. For NCA, due to computational reasons, 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 samples come from distributions with equal means. The smaller the p-value, the more significant the difference of the two average values is, and a p-value of 0.05 is considered statistically significant. As a reference, the classification errors of KNN on the clean data (without irrelevant features) and noisy data are also reported. From these experimental results, we arrive at the following observations:

(1) The performance of KNN is degraded significantly in the presence of irrelevant features, as reported in the literature.

(2) The original *banana* data is distributed in a two-dimensional space. This toy example provides us with an opportunity to examine the capability of each algorithm to extract nonlinear discriminant information by plotting the top two extracted features. We do not consider MMC here since MMC can extract only one feature. As can be seen from Fig. 3, the top two features extracted by LFE remove nearly all of the noise, and faithfully reconstruct the original data except for some minor distortions and rotation. While the original shape of *banana* is still recognizable from the result of NCA, it is not the case for both PCA and DANN. This is because some non-discriminant noise enters the top two features, leading to a severe distortion of the distribution of the original data.

(3) From Fig. 4, we observe that with respect to classification errors and the effectiveness

in reducing data dimensionality, LFE performs the best, DANN and NCA the second, MMC and PCA the worst. The feature extracted MMC clearly does not encode all of the useful information. The performance of DANN is conditioned heavily on the correct estimate 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).

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

(5) Both RELIEF and LFE can significantly improve the performance of KNN performed on noisy data. 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 **W** plotted in Fig. 5. 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.

(6) An important feature of LFE, compared to other feature extraction algorithms such as PCA and DANN, is that LFE is capable of eliminating irrelevant features, a nice property inherited from RELIEF. We observe in the plotted distance metric matrices in Fig. 5 that the lower right corner (the last 10 rows and columns) that corresponds to artificially added irrelevant features takes small values near zero. This becomes even more clearer when the diagonal elements of a distance metric matrix are plotted against the feature weights learned by RELIEF. The irrelevant features have been largely removed by LFE. This explains why LFE becomes flatten after reaching the minimum classification errors in almost all datasets (Fig. 4). However, the diagonal elements of **W** corresponding

to irrelevant features, though small, are not exactly equal to zero. This leads to a minor overfitting problem in some datasets (e.g., *ringnorm*). However, compared to PCA and DANN, overfitting in LFE is much alleviated.

#### 6.2 Experiments on USPS and Face Datasets

We then compare LFE with DANN, NCA, MMC and PCA on two real-world problems: the USPS handwritten digit recognition (*USPS*) dataset [23] and the AR face recognition (*face*) [24] dataset. These are two most studied object-recognition problems. Due to the high feature dimensionality, one of the main purposes of feature extraction in these studies, in addition to improving classification accuracy, is to reduce the precessing time in a classifier system. For the *USPS* dataset, we choose only two classes, namely "3" and "5", as they represent two of the most challenging digits to classify, and for the *face* dataset, the task is to classify female against male (see Fig. 2 for some sample images). Note that in the *face* dataset the data dimensionality significantly outnumbers the training samples. For both *face* and *USPS*, it is computationally infeasible to directly apply NCA to the original feature space. We therefore first perform PCA that projects data onto its leading 50 principal components and then perform NCA.

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. 6. 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. For both data sets, LFE reaches the minimum classification errors around 20 features, both DANN and PCA perform worse than LFE for nearly all feature numbers, and MMC is the worst one. NCA performs similarly to LFE on *USPS* but much worse on *face*. With 20 features, the classification errors (STDs) of LFE, NCA, DANN, and PCA for *USPS* are 2.0 (0.7)%, 2.3(0.8)%, 2.6 (0.9)% and 2.7 (1.0)%, respectively, and for *face* 8.0 (1.0)%, 11.8(1.8)%, 11.4 (0.7)% and 13.5 (1.3)%, respectively. For both data sets, the difference in performance between PCA and LFE is approximately equal to or much larger than one



Figure 5: Distance metric matrices **W** learned on (a) *banana*, (b) *splice*, (c) *waveform*, (d) *ringnorm*, and (e) *twonorm*. The lower right corner (the last 10 rows and columns) that corresponds to artificially added irrelevant features takes small values near zero. The diagonal elements of distance metric matrices are plotted against the feature weights learned by RELIEF.

error bar (p-value < 0.01).

In Table 3, we list the CPU time of LFE, DANN, NCA, MMC and LFE. We note that both LFE and DANN are computationally expensive than PCA and MMC. The added computational cost comes from the search for the nearest neighbors and the computation of PCA that projects data onto a subspace on which DANN and LFE are performed. Nevertheless, both DANN and LFE are computationally much more 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 (Table 2). In fact, any classification 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  $\delta$  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 huge computation involved in the estimation of the structural parameters of SVM<sup>1</sup>, we only perform SVM on the *face* dataset using the first 20 and 50 features. 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 noise resistance of SVM, LFE performs significantly better than PCA (p-value < 0.01).

<sup>&</sup>lt;sup>1</sup>The structural parameters of SVM include the kernel width and the regularization parameter. These parameters are determined by grid searching over 5 levels of kernel widths  $\{0.1, 0.2, 0.5, 1, 1.5\}$  and 5 levels of regularization parameters  $\{1, 10, 100, 500, 1000\}$  using 10-fold cross-validation on the training data for each feature extraction algorithms.



Figure 6: Classification errors of KNN on the USPS and face datasets.

#### 6.3 Experiments on Microarray Data

We finally compare LFE with DANN, MMC and PCA on eight microarray data. In the literature, feature selection and weighting are more commonly used for the purpose of identifying disease related genes [25, 26]. However, in this paper, we use microarray data to illustrate how our algorithm performs on real-world data with a wide range of sample sizes, data dimensionality and class labels (see Table 1). For all microarray data sets, the feature dimensionality is much larger than the number of samples. A more detailed description of these microarray data sets can be found in [27].

Due to the small sample size, the leave-one-out cross validation (LOOCV) method is used to evaluate the performance of each algorithm. As can be seen from the previous sections, NCA does not have much advantage over LFE in terms of performance and computational efficiency. Moreover, it is very computationally expensive to perform NCA with LOOCV. Hence, we omit NCA in the experiment. Unlike the data we consider in the previous experiments, all microarray data sets except for *DLBCL* are multiclass problems. Hence, MMC can extract more than one feature.

The classification errors of KNN as a function of the top 50 extracted features are presented in Fig. 7. We observe that with respect to classification performance and the effectiveness of reducing data dimensionality, LFE performs the best, MMC the second, DANN and PCA the worst. Since in each iteration of LOOCV, KNN classifies a test sample either correctly or incorrectly (i.e., 0 or 1), we are unable to perform a t-test to quantify the performance of each algorithm, as we did in the previous experiments. MMC performs very well in these small sample-size problems, which is consistent with the results reported in [15]. However, it is clear from the figure that MMC does not extract all of the discriminant information. For example, in *9-Tumors*, the classification error of MMC is 48% with 8 features compared to 40% for LFE with 15 features.

## 7 Conclusion

We have provided a very easy-to-understand interpretation for RELIEF that clearly explains its success in practical applications. The new interpretation motivated us to propose a new feature extraction algorithm referred to as LFE as a natural generalization of RELIEF. LFE collects discriminant information locally and results in an eigenvalue decomposition problems. Similar to PCA, LFE also has a clear physical meaning and can be implemented easily with a comparable computational cost. Unlike PCA, LFE has an explicit mechanism to remove irrelevant features, and thus can provide a very good generalization capability. We have experimentally demonstrated that LFE performs significantly better than PCA, NCA, MMC and DANN. Given the popularity of PCA and the superior performance of LFE, we believe that our algorithm should find widespread use in similar applications.

## References

- R. Kohavi and G. H. John, "Wrappers for feature subset selection," Artificial Intelligence, vol. 97, no. 1-2, pp. 273–324, 1997.
- [2] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119–1125, November 1994.



Figure 7: Classification errors of PCA, LFE, MMC and DANN on eight microarray datasets.

- [3] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," Journal of Machine Learning Research, vol. 3, pp. 1157 – 1182, 2003.
- [4] M. Hilario and A. Kalousis, "Approaches to dimensionality reduction in proteomic biomarker studies," *Briefings in Bioinformatics*, to appear.
- [5] K. Kira and L. A. Rendell, "A practical approach to feature selection," in Proc. 9th International Conference on Machine Learning. Morgan Kaufmann, 1992, pp. 249 – 256.
- [6] T. G. Dietterich, "Machine learning research: Four current directions," AI Magazine, vol. 18, no. 4, pp. 97–136, 1997.
- [7] Y. Sun and J. Li, "Iterative RELIEF for feature weighting," in *Proc. 23rd International Conference on Machine Learning*. ACM Press, 2006, pp. 913–920.
- [8] D. Wettschereck, D. W. Aha, and T. Mohri, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 273–314, 1997.
- [9] R. Gilad-Bachrach, A. Navot, and N. Tishby, "Margin based feature selection theory and algorithms," in *Proc. 21st International Conference on Machine Learning*. ACM Press, 2004, pp. 43–50.
- [10] E. K. P. Chong and S. H. Zak, An Introduction to Optimization. New York: John Wiley and Sons Inc., 2001.
- [11] L. Saul and S. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, June 2003.
- [12] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 607–615, June 1996.

- [13] Y. Sun, S. Todorovic, J. Li, and D. Wu, "Unifying error-correcting and output-code AdaBoost through the margin concept," in *Proc. 22nd International Conference on Machine Learning*. ACM Press, 2005, pp. 872–879.
- [14] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.
- [15] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 157–165, January 2006.
- [16] T. Hastie, R. Tibshirani, and J. H. Friedman, The Elements of Statistical Learning. New York: Springer-Verlag, 2003.
- [17] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in Advances in Neural Information Processing Systems, 2005, pp. 513–520.
- [18] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in Advances in Neural Information Processing Systems, 2006, pp. 1473-1480.
- [19] C. Domeniconi, P. Jing, and D. Gunopulos, "Locally adaptive metric nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1281–1285, September 2002.
- [20] C. Domeniconi, D. Gunopulos, and P. Jing, "Large margin nearest neighbor classifier," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 899–909, July 2005.
- [21] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html
- [22] I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," in Proc. European Conference on Machine Learning, 1994, pp. 171–182.

- [23] Y. LeCun, B. Boser, J. S. Denker, D. Hendersen, R. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [24] A. Martinez and R. Benavente, "The AR face database," Tech. Rep., 1998.
- [25] Y. Sun, S. Goodison, J. Li, L. Liu, and W. Farmerie, "Improved breast cancer prognosis through the combination of clinical and genetic markers," *Bioinformatics*, vol. 23, no. 1, pp. 30–37, January 2007.
- [26] Y. Sun, Y. Cai, and S. Goodison, "Combining nomogram and microarray data for predicting prostate cancer recurrence," in *Proc. 8th IEEE International Conference* on *Bioinformatics and Bioengineering*, in press.
- [27] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis," *Bioinformatics*, vol. 21, no. 5, pp. 631–43, 2005.