IMMIGRATE: A Margin-based Feature Selection Method with Interaction Terms

Ruzhang Zhao
Tsinghua University

Pengyu Hong*
Brandeis University

Jun S. Liu*
Harvard University

Abstract

By balancing margin-quantity maximization and margin-quality maximization, the proposed IMMIGRATE algorithm considers both local and global information when using margin-based frameworks. We here derive a new mathematical interpretation of margin-based cost function by using the quadratic form distance (QFD) and applying both the large-margin and max-min entropy principles. We also design a new principle for classifying new samples and propose a Bayesian framework to iteratively minimize the cost function. We demonstrate the power of our new method by comparing it with 16 widely used classifiers (e.g. Support Vector Machine, k-nearest neighbors, RELIEF, etc.) including some classifiers that are capable of identifying interaction terms (e.g. SODA, hierNet, etc.) on synthetic dataset, five gene expression datasets, and twenty UCI machine learning datasets. Our method is able to outperform other methods in most cases.

1 INTRODUCTION

Feature selection is one of the most fundamental problems in machine learning (Fukunaga, 2013). The RELIEF algorithm by Kira and Rendell (1992) has been proven to be one of the most successful feature selection algorithms for its simplicity and effectiveness. It is interpreted to follow the large-margin principle and shown to be an online learning algorithm that solves a convex optimization problem with a margin-based cost function. Compared with exhaustive or heuristic combinatorial searches, RELIEF decomposes a complex, global and nonlinear classification task into a simple and local one. RELIEF is the first algorithm that uses hypothesis-margin to calculate feature weights for the purpose of classification. See Gilad-Bachrach et al. (2004) for a formal definition of the hypothesis-margin. Early RELIEF-based algorithms share the same idea that a margin is defined using the fixed 1-nearest-neighbor. Kononenko (1994) extended RELIEF to RELIEF-F, which uses multiple nearest neighbours to adjust feature weights. Gilad-Bachrach et al. (2004) proposed Simba, which updates the nearest neighbours every time the feature weights are updated. A few competitive feature selection methods have also been proposed based on the large hypothesis-margin principle (Crammer et al., 2003; Sun and Li, 2006; Yang et al., 2008). Another appealing property of the RELIEF-based algorithms is that they allow the examination of the knowledge associated with the nearest neighbors, which may not be reflected by the individual features.

Different from previous margin-based algorithms, which mainly focus on maximizing the margin-quantity (to obtain the largest margin) and ignore the robustness of the contributors of margin (i.e., margin-quality), Bei and Hong (2015) recognized the importance of margin-quality and proposed "IM4E", which incorporates the max-min entropy principle into the margin-based cost function to take the quality of margin into consideration. Although the RELIEF-based algorithms indirectly consider the interactions among features by normalizing the feature weights, it is notable that feature weights from RELIEF do not reflect the natural effects of association. For example, RELIEF and many of its extensions do not tell us whether the cause of a high feature weight is from its linear effect or its interaction with other features (Urbanowicz et al., 2018). This problem is partially addressed in LFE (Sun and Wu, 2008), where a weight matrix is used to replace the weight vector so that the effects of interaction terms can be reflected to some extent.

Motivated by the above approaches, we propose the *Iterative Max-MIn entropy marGin-maximization with* *inteRAction TErms* (IMMIGRATE, henceforth) algorithm which delivers the following novelties. First, we propose a quadratic form distance based framework for capturing interaction terms. Second, we take the robustness of margin (margin quality) into account to derive a margin-based cost function. Third, we design a novel prediction method to use the weight matrix effectively. Fourth, we propose a Bayesian framework for the iterative optimization. We also find a closed-form solution to the Bayesian framework with a convergence guarantee. Our experimental results show that IMMIGRATE outperforms many popular classifiers (including a few classifiers that consider interaction terms) on UCI machine learning datasets. Finally, several options of the IMMIGRATE algorithm, such as the removal of small weights to obtain sparse results, are also considered. We design the IM4E-IMMIGRATE algorithm for effectively selecting interaction terms in high-dimensional datasets. Gene expression datasets were used to demonstrate the effectiveness of the IM4E-IMMIGRATE algorithm on high-dimensional datasets.

The rest of the paper is organized as follows. Section 2 explains the mathematical foundation of the RELIEF-based algorithms. The IMMIGRATE algorithm is proposed in Section 3. Section 4 summarizes and discusses the experiments on synthetic dataset, gene expression datasets and UCI datasets. The paper is concluded with discussions in Section 5.

2 RELIEF

The RELIEF algorithm (Kira and Rendell, 1992) provides a framework for using a fixed number of instances to calculate feature weights. Feature weights are usually referred to as feature "scores" and range from -1 to 1. After calculating the feature weights, a certain threshold is set to select relevant features and discard irrelevant features. RELIEF can be viewed as a convex optimization problem with a cost function whose mathematical expression is shown in Eq. 2.1 if the threshold is set to be 0. That is, RELIEF minimizes

$$C = \sum_{n=1}^{M} \left(\vec{w}^T \left| \vec{x}_n - NH(\vec{x}_n) \right| - \vec{w}^T \left| \vec{x}_n - NM(\vec{x}_n) \right| \right), \quad subject\ to\ :\ \vec{w} \geq 0\ and\ \|\vec{w}\|_2^2 = 1, \tag{2.1}$$

where $NH(x)$ is the nearest "hit" (from the same class) of x; $NM(x)$ is the nearest "miss" (from a different class) of x; $\left|\vec{x}_n - NH(\vec{x}_n)\right|$ calculates the absolute element-wise differences; $\vec{w}^T\left|\vec{x}_n - NH(\vec{x}_n)\right|$ is the weighted Manhattan distance. If we denote $\vec{u} = \sum_{n=1}^{M} \left(\left|\vec{x}_n - NH(\vec{x}_n)\right| - \left|\vec{x}_n - NM(\vec{x}_n)\right|\right)$, Eq. 2.1 can be simplified as $C = \vec{w}^T\vec{u}$. Minimizing C can be solved using the Lagrange multiplier method with the Lagrangian $L = \vec{w}^T\vec{z} + \lambda(\|\vec{w}\|_2^2 - 1) + (-\vec{w}^T)\vec{\theta}$, where $\lambda \geq 0$ and $\vec{\theta} \geq 0$. Using the Karush-Kuhn-Tucker condition by Kuhn and Tucker (2014), the solution is straightforward: $\vec{w} = (-\vec{u})^+ / \|(-\vec{u})^+\|_2$, where $(\vec{a})^+ = [max(a_1, 0), max(a_2, 0), \cdots, max(a_A, 0)]$. This solution to is fundamental to understanding the RELIEF-based algorithms. The pseudo code of the original RELIEF algorithm is listed in Algorithm 1.

Algorithm 1 Original RELIEF Algorithm

```
State : Feature selection for binary classification
$N \leftarrow$ number of training instances
$A \leftarrow$ number of features(i.e. attributes)
$M \leftarrow$ number of randomly chosen instances out of $M$ to update weight $\vec{w}$
initial all features weights to 0 $\vec{w} := 0$
  for $i := 1$ to $M$ do
    randomly select a "target" instance $x_i$
    find its $NH(x_i)$ and $NM(x_i)$ in Eq. 2.1
    for $a := 1$ to A do
      $w[a] := w[a] - (x_i[a] - NH(x_i)[a])^2/M + (x_i[a] - NM(x_i)[a])^2/M$
    end for
  end for
  return the vector $W$ of feature scores that estimate the quality of features
```

3 IMMIGRATE

We will establish the framework of IMMIGRATE algorithm in this section. Since IMMIGRATE is an extension of IM4E algorithm, the derivation of IM4E is briefly discussed to help the explanation of IMMIGRATE.

Let $\mathcal{D} = \{z_n | z_n = (\vec{x}_n, y_n) \in \mathcal{R}^{A+1}, \vec{x}_n \in \mathcal{R}^A, y_n \in \{-1, 1\}\}_{n=1}^N$, where N is the number of instances; A is the number of features; $\vec{x}_n$ represents feature vector and y_n is class. We only consider binary classification in our formulation. Because binary classification is the basic classification task in machine learning and is often used to test the performance of feature selection algorithms. The definitions in our margin-based framework (e.g., hits and misses) make it easy to extend a binary classification formulation to a multiple classification problem. Our IMMIGRATE implementation in R works for multiple classification tasks. Here, we define the following notations $\mathcal{H}_n = \{j \in \{1, 2, \cdots, N\} | z_j \in \mathcal{D}, y_j = y_n\ and\ j \neq n\}$, $\mathcal{M}_n = \{j \in \{1, 2, \cdots, N\} | z_j \in \mathcal{D}, y_j \neq y_n\}$, where $\mathcal{H}_n$ and $\mathcal{M}_n$ represent the index sets of hits and misses of the instance z_n, respectively.

3.1 Max-Min Entropy Principle

Given a generalized distance metric $d(i,j)$ between two instances z_i and z_j, a hypothesis-margin (Gilad-Bachrach et al., 2004) is defined as

$$\rho_{n,h,m} = d(n,m) - d(n,h), \tag{3.2}$$

where $z_h, h \in \mathcal{H}_n$ and $z_m, m \in \mathcal{M}_n$ represent the nearest hit and nearest miss for instance z_n. Since the generalized distance metric is unknown because of the unknown feature weights, the nearest hit and nearest miss are undetermined under such a framework. Hence, we adopt the method proposed in Sun and Li (2006), Sun and Wu (2008), Sun et al. (2010) and Bei and Hong (2015), which defines the margin as follows.

$$\rho_n = \sum_{m\in\mathcal{M}_n} \beta_{n,m} d(n,m) - \sum_{h\in\mathcal{H}_n} \alpha_{n,h} d(n,h) \tag{3.3}$$

where $\alpha_{n,h} \geq 0$, $\beta_{n,m} \geq 0$, $\sum_{h\in\mathcal{H}_n} \alpha_{n,h} = 1$, $\sum_{m\in\mathcal{M}_n} \beta_{n,m} = 1$, for $\forall n \in \{1, \cdots, N\}$. As in the above design, hidden random variable $\alpha_{n,h}$ represents the probability that $z_{n,h}$ is the nearest hit of instance z_n, while hidden variable $\beta_{n,m}$ indicates the probability that $z_{n,m}$ is the nearest miss of instance z_n. The derivations of $\alpha_{n,h}$ and $\beta_{n,m}$ are closely related to the generalized distance metric.

As proposed, the probabilities $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$ represent some distribution of hits and misses. The robustness of the margin of z_n can be defined using its hit probabilities $\{\alpha_{n,h}\}$ and miss probabilities $\{\beta_{n,m}\}$. Thus, the hit entropy and miss entropy are respectively defined as $E_{hit} = -\sum_{h\in\mathcal{H}_n} \alpha_{n,h} \log \alpha_{n,h}$ and $E_{miss} = -\sum_{m\in\mathcal{M}_n} \beta_{n,m} \log \beta_{n,m}$.

The following two scenarios help to explain the intuition of using the hit entropy and miss entropy. In scenario A, all neighbours are distributed evenly around the target instance. In scenario B, the neighbour distribution is highly uneven. In particular, one instance is quite close to the target and the rest are quite far away from the target. An easy experiment to test the robustness of a margin is to discard one instance from our system and then check how much the system will change. Firstly, in scenario A, if the nearest hit is discarded, the margin will not change much since there are many other hits evenly distributing around target. However, in scenario B, the disappearance of the nearest hit will largely reduce the margin since its hit probabilities concentrate at a few hits. Thus, hit probabilities prefer large entropy, such as scenario A. Meanwhile. The miss probabilities prefer small entropy (e.g., scenario B) because the disappearance of the nearest miss will largely increase the margin. In the max-min entropy framework Bei and Hong (2015), one should maximize the hit entropy and minimize the miss entropy. This max-min entropy principle is an extension of the large-margin principle and optimizes the hit entropy and miss entropy to be consistent with the large-margin principle.

3.2 IM4E Algorithm

The IM4E algorithm Bei and Hong (2015) was developed under the framework Eq. 3.3 to consider margin-quality so as to improve the robustness of the contributors of margin. The max-min entropy principle was designed to reduce the distributions to the large margin criterion caused by noise and sampling variations. Compared with similar algorithms I-RELIEF(Sun and Li, 2006) and Local Learning(Sun et al., 2010), which were both intuitively designed, IM4E algorithm had a strong mathematical foundation.

In framework 3.3, for IM4E, the generalized distance metric was chosen to be weighted Manhattan distance. The cost function (Eq. 3.4) used by IM4E balances both margin-quantity maximization and margin-quality maximization. The iterative algorithm for IM4E (including Eq. 3.5 for updating the hit and miss probabilities and Eq. 3.6 for updating the feature weights with $\lambda = 1$) was naturally derived to minimize C from its cost function Eq. 3.4.

$$\begin{aligned} C = \vec{w}^T \sum_{n=1}^{N} \Bigg(\sum_{h\in\mathcal{H}_n} \alpha_{n,h} |\vec{x}_n - \vec{x}_h| - \sum_{m\in\mathcal{M}_n} \beta_{n,m} |\vec{x}_n - \vec{x}_m| \Bigg) \\ + \sigma \sum_{n=1}^{N} [E_{miss}(z_n) - E_{hit}(z_n)] + \lambda \|\vec{w}\|^2, \\ \textit{subject to} \ : \ \vec{w} \geq 0 \textit{ and } \|\vec{w}\|_1 = 1, \end{aligned} \tag{3.4}$$

where $E_{miss}(z_n) = -\sum_{m\in\mathcal{M}_n} \beta_{n,m} \log \beta_{n,m}$, $E_{hit}(z_n) = -\sum_{h\in\mathcal{H}_n} \alpha_{n,h} \log \alpha_{n,h}$ and σ, λ are both tune parameters.

$$\begin{aligned} \alpha_{n,h} = exp(\frac{-f(\vec{x}_n, \vec{x}_h)}{\sigma}) / \sum_{j\in\mathcal{H}_n} exp(\frac{-f(\vec{x}_n, \vec{x}_j)}{\sigma}), \\ \beta_{n,m} = exp(\frac{-f(\vec{x}_n, \vec{x}_m)}{\sigma}) / \sum_{k\in\mathcal{M}_n} exp(\frac{-f(\vec{x}_n, \vec{x}_k)}{\sigma}), \end{aligned} \tag{3.5}$$

where $f(\vec{x}_n, \vec{x}_h) = \vec{w}^T |\vec{x}_n - \vec{x}_h|$ is the weighted Manhattan Distance.

$$\begin{aligned} \vec{v} = \sum_{n=1}^{N} \Bigg(\sum_{m\in\mathcal{M}_n} \beta_{n,m} |\vec{x}_n - \vec{x}_m| - \sum_{h\in\mathcal{H}_n} \alpha_{n,h} |\vec{x}_n - \vec{x}_h| \Bigg), \\ \vec{w} = \vec{v}^+ / \|\vec{v}^+\|_1, \end{aligned} \tag{3.6}$$

where $\vec{v}^+ = [max(v_1, 0), max(v_2, 0), \cdots, max(v_p, 0)]$.

The pseudo code of IM4E algorithm is listed in Algorithm 2.

Algorithm 2 IM4E Algorithm

```
  State : Feature selection for binary classification
  $N$ and $A$ have the same definitions in Algorithm 1
  Input : a training dataset $\{z_n = (\vec{x}_n, y_n)\}_{n=1,\cdots,N}$
Let t = 0, randomly initialize $\vec{w}^{(0)} > 0$ satisfying the sum of feature weights equal to 1
Repeat
  t=t+1
  Calculate $\{\alpha^{(t)}_{n,h}\}$ and $\{\beta^{(t)}_{n,m}\}$ using Eq. 3.5 respectively
  Calculate $\vec{w}^{(t)}$ using Eq. 3.6
Until the change of C is small enough or the iteration indicator $t$ reaches a preset limit
Return $\vec{w}^{(t)}$
```

3.3 imIM4E Algorithm

Imbalance data often occurs in real applications and causes significant problems to various machine learning techniques. For example, in biomedical domain, the patient instances are often much less than the normal instances. Our experiences with the marge-based machine learning techniques invoked us to develop a new method to balancing the margin contributions of positive class instances and negative class instances, which we term the margin imbalance problem. To this end, we have formulated a new algorithm, called imbalance IM4E(imIM4E), which represents IM4E considering margin imbalance. The solution to the margin imbalance is adding a hyper-parameter ρ to balance the difference in class sizes, and change the cost function of imIM4E algorithm to

$$\begin{aligned} C = &\Big(\sum_{n\in +} + \rho \sum_{n\in -}\Big) \\ &\Big\{\vec{w}^T\Big(\sum_{h\in\mathcal{H}_n}\alpha_{n,h}|\vec{x}_n - \vec{x}_h| - \sum_{m\in\mathcal{M}_n}\beta_{n,m}|\vec{x}_n - \vec{x}_m|\Big) \\ &+ \sigma[E_{miss}(z_n) - E_{hit}(z_n)]\Big\} + \lambda\|\vec{w}\|^2, \\ &\quad \textit{subject to} \;:\; \vec{w} \geq 0 \textit{ and } \|\vec{w}\|_1 = 1, \end{aligned} \tag{3.7}$$

where $n \in +$ and $n \in -$ indicate that z_n is from the positive class or the negative class, respectively.

The optimization of imIM4E's cost function shares similar steps with IM4E. Step 1: Start with an initial weight vector $\vec{w}$(randomly initialize $\vec{w}$ making $\vec{w} \geq 0$, and $\|\vec{w}\|_1 = 1$). Step 2: Fix $\vec{w}$, update $\{\alpha_{n,h}\}$ and $\{\beta_{n.m}\}$ by Eq. 3.5. Step 3: Fix $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$, update $\vec{w}$ by Eq. 3.6 with ρ included. Start with Step 1, the optimization algorithm iteratively executes Steps 2 and 3 until convergences. The hyper-parameters σ and ρ are tuned by cross-validation (with normalization, the parameter λ does not matter as long as $\lambda \neq 0$). The proof of convergence is the same as the one in Bei and Hong (2015).

The hyper-parameter ρ is not used in making prediction with the learned feature weights. IM4E and imIM4E share the same prediction function Eq. 3.8.

$$y' = \arg\min_c \sum_{y_n=c} P^c_n f(\vec{x}_n, \vec{x}\,'), \tag{3.8}$$

where c denotes the class, and

$$P^c_n = \frac{exp\big(-f(\vec{x}_n, \vec{x}\,')/\sigma\big)}{\sum_{y_k=c} exp\big(-f(\vec{x}_k, \vec{x}\,')/\sigma\big)}, \tag{3.9}$$

where $f(\vec{x}_n, \vec{x}\,') = \vec{w}^T|\vec{x}_n - \vec{x}\,'|$.

3.4 IMMIGRATE Algorithm

To capture interactions among features, we developed the IMMIGRATE algorithm by choosing the generalized distance metric in framework Eq. 3.3 to be a quadratic form distance (the Mahalanobis distance). The "IMMIGRATE" stands for **I**terative **M**ax-**MI**n entropy mar**G**in-maximization with inte**RA**ction **TE**rms Algorithm.

The quadratic form distance for instances z_i and z_j with a weight matrix $\mathbf{W}$ is defined as

$$d(i,j) = |\vec{x}_i - \vec{x}_j|^T \mathbf{W} |\vec{x}_i - \vec{x}_j|. \tag{3.10}$$

The weight matrix is a natural extension of the weight vector since a weight vector can be represented as a diagonal matrix. The cost function Eq. 3.11 was designed to maximize the generalized margin under max-min entropy principle.

$$\begin{aligned} C = &\sum_{n=1}^{N}\Big(\sum_{h\in\mathcal{H}_n}\alpha_{n,h}|\vec{x}_n - \vec{x}_h|^T\mathbf{W}|\vec{x}_n - \vec{x}_h| \\ &- \sum_{m\in\mathcal{M}_n}\beta_{n,m}|\vec{x}_n - \vec{x}_m|^T\mathbf{W}|\vec{x}_n - \vec{x}_m|\Big) \\ &+ \sigma\sum_{n=1}^{N}[E_{miss}(z_n) - E_{hit}(z_n)], \\ &\quad \textit{subject to} \;:\; \mathbf{W} \geq 0 \textit{ and } \|\mathbf{W}\|_F^2 = 1, \\ &\quad \sum_{h\in\mathcal{H}_n}\alpha_{n,h} = 1\;,\; \sum_{m\in\mathcal{M}_n}\beta_{n,m} = 1\;\forall\; n, \end{aligned} \tag{3.11}$$

where $\|\mathbf{W}\|_F^2$ is the Frobenius norm of $\mathbf{W}$. $\|\mathbf{W}\|_F^2 = \sum_{i,j} w_{i,j}^2 = \sum_i \lambda_i^2$, with λ_i's are eigenvalues of matrix $\mathbf{W}$.

Now we propose a Bayesian framework including three steps to iteratively minimize the cost function. The framework starts from a randomly generated weight matrix as prior one and ends up until the change of cost reaches a preset limit, and then the posterior one is used as final output.

Step 1: The optimization of cost function Eq. 3.11 starts from a randomly initialized $\mathbf{W}$ (satisfying $\mathbf{W} \geq 0$ and $\|\mathbf{W}\|_F^2 = 1$). Then following two steps are iterated to minimize the cost function. Step 2: Fix $\mathbf{W}$, update $\{\alpha_{n,h}\}$ and $\{\beta_{n.m}\}$. Step 3: Fix $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$, update $\mathbf{W}$.

3.4.1 Fix W, Update $\{\alpha_{n,h}\}$ and $\{\beta_{n.m}\}$

Fixing $\mathbf{W}$, we can update $\alpha_{n,h}$ and $\beta_{n,m}$ as Eq. 3.5, where $f(\vec{x}_n, \vec{x}_h) = \left|\vec{x}_n - \vec{x}_h\right|^T \mathbf{W} \left|\vec{x}_n - \vec{x}_h\right|$.

The derivative of the cost function with respect to $(\alpha_{n,h}, \beta_{n,m})$ is

$$\frac{\partial^2 C}{\partial(\alpha_{n,h}, \beta_{n,m})} = \begin{pmatrix} \sigma/\alpha_{n,h} & \partial^2 C/\partial\beta_{n,m}\alpha_{n,h} \\ \partial^2 C/\partial\beta_{n,m}\alpha_{n,h} & -\sigma/\beta_{n,m} \end{pmatrix}, \tag{3.12}$$

$$\left|\frac{\partial^2 C}{\partial(\alpha_{n,h}, \beta_{n,m})}\right| = -\frac{\sigma^2}{(\alpha_{n,h}\beta_{n,m})} - (\frac{\partial^2 C}{\partial\beta_{n,m}\alpha_{n,h}})^2 < 0. \tag{3.13}$$

Therefore, when fixing $\mathbf{W}$, a saddle point in $(\alpha_{n,h}, \beta_{n,m})$ space can be found.

3.4.2 Fix $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$, Update W

Fixing $\alpha_{n,h}$ and $\beta_{n,m}$, the derivation of $\mathbf{W}$ is a little computationally arduous. However, we can derive a closed form solution for $\mathbf{W}$.

Theorem 3.1 *Fixing $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$, the cost function Eq. 3.11 has a closed-form solution for updating $\boldsymbol{W}$.*

$$\mathbf{\Sigma} = \sum_{n=1}^{N} \Sigma_{n,H} - \Sigma_{n,M}, \ \mathbf{\Sigma}\ \psi_i = \mu_i\ \psi_i, \tag{3.14}$$

where $\Sigma_{n,H} = \sum_{h\in\mathcal{H}_n} \alpha_{n,h}\left|\vec{x}_n - \vec{x}_h\right|\left|\vec{x}_n - \vec{x}_h\right|^T$, $\Sigma_{n,M} = \beta_{n,m}\left|\vec{x}_n - \vec{x}_m\right|\left|\vec{x}_n - \vec{x}_m\right|^T$, *and* $\|\psi_i\|_2^2 = 1$, $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_A$. *$\psi_i$'s and μ_i's are the eigenvectors and eigenvalues of $\mathbf{\Sigma}$ separately.*

$$\mathbf{W} = \Phi\,\Phi^T, \tag{3.15}$$

where $\Phi = (\sqrt{\eta_1}\psi_1, \sqrt{\eta_2}\psi_2, \cdots, \sqrt{\eta_A}\psi_A)$, $\sqrt{\eta_i} = \sqrt{(-\mu_i)^+ / \sqrt{\sum_{i=1}^{A}((-\mu_i)^+)^2}}$.

The proof of Theorem 3.1 is in supplementary material.

```
Algorithm 3 IMMIGRATE Algorithm
  State : Feature selection for binary classification
  $N$ & $A$ have the same meaning in Algorithm 1
  Input : a training dataset $\{z_n = (\vec{x}_n, y_n)\}_{n=1,\cdots,N}$
Let t = 0, randomly initialize $\mathbf{W}$ satisfying $\mathbf{W} \geq 0$, and $\|\mathbf{W}\|_F^2 = 1$
Repeat
  t=t+1
  Calculate $\{\alpha_{n,h}^{(t)}\}$ and $\{\beta_{n,m}^{(t)}\}$ using Eq. 3.5 with quadratic form distance
  Calculate $\mathbf{W}^{(t)}$ using theorem 3.1 and Eq. 3.15
Until the change of C is small enough or the iteration indicator $t$ reaches a preset limit
Return $\mathbf{W}^{(t)}$
```

3.4.3 Small Weights Removed

When there are too many features and interaction terms, it is easy to cause overfitting. Thus, we added an option in IMMIGRATE algorithm to remove small weights. In the Bayesian framework, Step 4 was added: Set weights in $\mathbf{W}$ which are smaller than a preset threshold "t" to be 0, and normalize the $\mathbf{W}$. Thus, Step 2,3,4 are iterated to minimize the cost function.

3.4.4 Classify New Samples

We improve the prediction method developed in IM4E (Bei and Hong, 2015) to use the learned weight matrix $\mathbf{W}$.

For a new sample $z' = (\vec{x}', y')$, we use the learned weight matrix "$\mathbf{W}$" to select a class for z' using Eq. 3.8 and Eq. 3.9, where $f(\vec{x}_n, \vec{x}') = \left|\vec{x}_n - \vec{x}'\right|^T \mathbf{W} \left|\vec{x}_n - \vec{x}'\right|$.

Here, we classify the new samples into the class whose expected distance is the shortest. In fact, this is still a nearest-neighbor method for classification where the distance metric is a generalized one to better capture interactions between features.

4 EXPERIMENTS

All the algorithms used in our experiments either have implementations in R or have been implemented by us in R(R package "Immigrate", available on github), such as Simba (Gilad-Bachrach et al., 2004), LFE (Sun and Wu, 2008), IM4E (Bei and Hong, 2015), imIM4E, and IMMIGRATE.

4.1 Results on Synthetic Dataset

In this experiment, we tested the robustness of LFE and IMMIGRATE using synthesized dataset where two features were purposely designed for interaction term selection. We call a feature selection algorithm robust if the results obtained from original dataset are consistent with the ones from datasets with noises. We generated dataset1 with 200 samples as follows. 100 samples with class 0 and 100 samples with class 1 were randomly generated from Gaussian distributions with mean [4,2], variance diag[1,1] and with mean[6,0], variance diag[1,1] separately. Noises with class 0 and class 1 were randomly generated from a Gaussian distribution with mean [8,-2], variance diag[8,8] and with mean[2,4], variance diag[8,8], respectively. The scatter plot of dataset with 10% noise is shown in Fig. 1. The noises were designed to disturb the detection of the interaction term. The level of noises started from 0.05, and was gradually increased by 0.05 each time until it was larger than 0.5. We ran both LFE and IMMIGRATE(IGT) on the synthesized dataset and collected the weights corresponding to the interaction term between features 1 and 2. The interaction weights learned by LFE and IMMIGRATE are compared in Fig. 2, which clearly shows that IMMIGRATE is highly robust while LFE is less stable and is prone to noise.

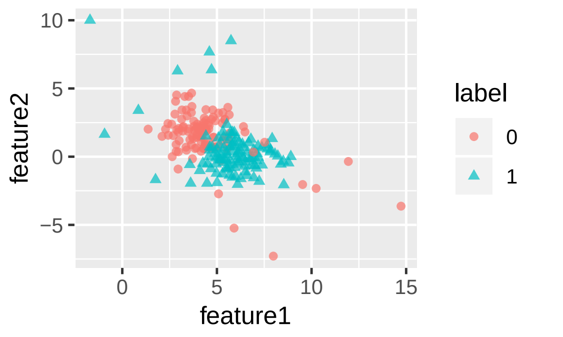

Figure 1: Synthesized Dataset1 with 10% noise.

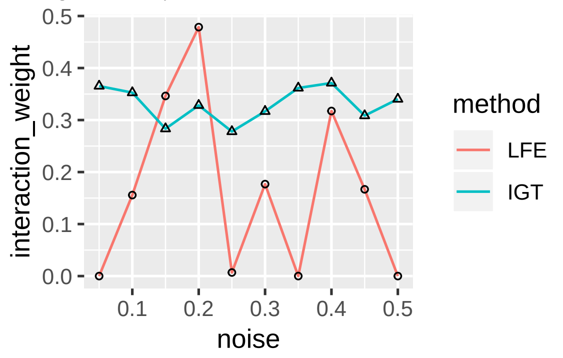

Figure 2: IMMIGRATE is more robust than LFE in learning the weights, which indicate the interaction between two features, from noisy datasets.

4.2 Results on Real Datasets

We also compared imIM4E and IMMIGRATE with several existing popular algorithms using many real datasets obtained from the UCI machine learning database. Cross validation was used to test the performance. The following existing algorithms were used in this experiment: Support Vector Machine (Soentpiet et al., 1999) with Sigmoid Kernel (SV1), with Radial basis function Kernel (SV2), LASSO (Tibshirani, 1996) (LAS), Decision Tree (Freund and Mason, 1999) (DT), Naive Bayes Classifier (John and Langley, 1995) (NBC), Radial basis function Network (Haykin, 1994) (RBF), 1-Nearest Neighbor (Aha et al., 1991) (1NN), 3-Nearest Neighbor (3NN), RELIEF (Kira and Rendell, 1992) (REL), RELIEF-F (Kononenko, 1994; Robnik-Šikonja and Kononenko, 2003) (RFF), Simba (Gilad-Bachrach et al., 2004) (SIM), Linear Discriminant Analysis (Fisher, 1936) (LDA), Adaptive Boosting (Freund et al., 1996) (ADB). In addition, several methods designed for detecting interaction terms were included: LFE (Sun and Wu, 2008), Stepwise conditional likelihood variable selection for Discriminant Analysis (Li and Liu, 2018) (SOD), hierNet (Bien et al., 2013) (HIN). When reporting the experimental results, we indicate IM4E as IM4, imIM4E as imI, IMMIGRATE as IGT, "IMMIGRATE with small weights removed" as IGS and IM4E-IMMIGRATE as EGT.

The settings of these methods were chosen as they were suggested in original papers: RELIEF and Simba used Euclidean distance and 1-NN classifier; RELIEF-F used Manhattan distance and k-NN classifier(k=1,3,5 was decided by internal cross-validation); in SODA, gam(=0,0.5,1) was determined by internal cross-validation and logistic regression was used for prediction. The IM4E algorithm owns two hyper-parameters λ and σ, where we fixed λ as 1 since it has no actual contribution and tuned σ as suggested in Bei and Hong (2015). The imIM4E algorithm has three hyper-parameters λ, σ and ρ, where λ and σ were dealt with just as IM4E algorithm did. When tuning ρ, we fixed σ. ρ started from $\rho_0 = 1.1$ and was gradually reduced by 0.02 each time until it was smaller than 0.9. The IMMIGRATE algorithm has one hyper-parameter σ which was tuned as IM4E did. The preset threshold in IM4E-IMMIGRATE was 1/A, where A is the number of features.

4.2.1 Results on Gene Expression Datasets

Gene expression data typically has thousands of features. For high-dimensional datasets, IMMIGRATE is not computationally effective because of the quadratic distance metric. IM4E-IMMIGRATE algorithm was designed to facilitate IMMIGRATE in

Table 1: Summarizes the accuracies on five high-dimensional gene expression datasets.

Data	SV1	SV2	LAS	DT	NBC	1NN	3NN	SOD	IM4	**EGT**
GLI	85.13	85.97	85.23	83.84	82.96	88.70	87.69	88.70	87.54	89.12
COL	73.69	81.98	80.64	69.16	71.12	72.08	77.93	78.06	84.28	78.57
ELO	72.91	90.22	74.59	77.29	76.31	85.56	91.34	86.89	88.95	88.56
BRE	75.98	88.69	91.35	76.42	69.40	82.97	73.59	82.59	88.13	90.17
PRO	71.26	69.89	87.86	86.42	67.96	83.18	82.74	83.24	88.02	89.52
W,T,L[1]	0,0,5	2,0,3	2,0,3	0,0,5	0,0,5	0,0,5	1,0,4	0,2,3	1,1,3	-,-,-

[1] The last row shows the number of times each method W,T,L (win,tie,loss) compared with IM4E-IMMIGRATE(**EGT**) by paired t-test.

high-dimensional datasets. Firstly, apply IM4E to obtain a weight vector. Then, select a certain number of features based on the weight vector learned by IM4E, and initialize the weight matrix of IMMIGRATE to be consistent with the weight vector learned by IM4E. Comparison on five publicly available gene expression datasets were carried out: GLI(Freije et al., 2004), Colon(Alon et al., 1999)(COL), Myeloma(Tian et al., 2003)(ELO), Breast(Van't Veer et al., 2002)(BRE), Prostate(Singh et al., 2002)(PRO). All five datasets have less than 180 samples and more than ten thousand features. Feature selection methods are widely tested in these high-dimensional datasets.

We performed 10-fold cross-validation for ten times, which means 100 trials were carried out. We reported the average accuracies on the corresponding datasets in Table 1. The last row "(W,T,L)" indicates the number of times each algorithm W,T,L (win,tie,loss) when compared with IM4E-IMMIGRATE (EGT) using the paired Student's t-test with the significance level set to be $\alpha = 0.05$. We say an algorithm A is significantly better than (i.e. win) another algorithm B on a dataset C if the p-value of the paired Student's t-test with corresponding null hypothesis is less than $\alpha = 0.05$.

As shown in Table 1, although IM4E-IMMIGRATE (EGT) is not always the best, it outperforms other methods in most cases. In particular, EGT beats IM4E three times out of five datasets.

4.2.2 Results on UCI Datasets

UCI datasets from Frank and Asuncion (2010) were used to compare the performance of a cohort of algorithms. The datasets include Breast Cancer (BRC), Wholesale customers (CUS), MONK's Problems (MON), Wine (WIN), Cryotherapy (CRY), Haberman's Survival (HMS), Connectionist Bench (Sonar, Mines vs. Rocks) (SMR), Urban Land Cover (URB), Immunotherapy Dataset (IMM), User Knowledge Modeling (USE), Vertebral Column (VEC), Ecoli (ECO), Glass Identification (GLA), Statlog (Heart) (STA), Ionosphere (ION), Lymphograph (LYM), Parkinsons (PAR),Breast Cancer Wisconsin (Prognostic) (BCW), Wine Quality (WQT) and Pima-Indians-Diabetes (PID). For datasets with more than two classes, the largest two classes were used in this experiment. Also, features in all datasets were normalized before being used.

We also performed 10-fold cross-validation for ten times. We reported the average accuracies on the corresponding datasets in Tables 2 and 3. The last row "(W,T,L)" indicates the number of times each algorithm W,T,L (win,tie,loss) when compared with IMMIGRATE(IGT) using the paired Student's t-test with the significance level set to be $\alpha = 0.05$.

Although IMMIGRATE is not always the best, it outperforms other methods in most cases. The imIM4E could improve the performance of IM4E slightly but not significantly. This is because we have searched ρ in the range of [0.9, 1.1]. When $\rho = 1$, imIM4E is just IM4E itself. Hence, imIM4E performs comparably to IM4E in most cases. The last column in Table 2 lists the results of IMMIGRATE with small weights removed. On some datasets where IMMIGRATE did not perform well, for example, on the "ION" dataset, IGS outperformed IGT. We suspect that this situation was caused by overfitting and IGS can reduce the number of chosen weights significantly to reduce overfitting.

5 CONCLUSION & DISCUSSION

In this paper, we proposed a novel feature selection algorithm IMMIGRATE for detecting interaction terms. A Bayesian framework was designed for implementing the IMMIGRATE algorithm and the close-form of solution was derived in Theorem 3.1. Large-margin and max-min entropy principles were used to present a generalized framework for feature learning and IMMIGRATE was a natural extension. We proposed non-linear margin-based cost function and minimized the cost function iteratively. IMMIGRATE was also extended to work on high-dimensional datasets via

Table 2: Summarizes the accuracies on UCI datasets.

Data	SV1	SV2	LAS	DT	NBC	RBF	1NN	3NN	REL	RFF	SIM	LFE	LDA	**IGT**	IGS
BRC	65.14	75.49	72.87	68.86	70.34	64.04	68.13	68.98	56.02	58.62	62.21	65.00	72.01	75.22	74.08
CUS	86.45	88.86	89.61	89.61	89.45	86.78	86.48	88.66	82.05	84.66	84.34	86.36	90.25	90.05	89.37
MON	74.39	91.65	74.99	86.44	74.02	68.19	75.10	84.36	61.42	61.81	64.96	64.78	74.40	99.51	95.43
WIN	98.31	99.31	98.62	93.08	97.31	97.23	96.38	96.62	87.23	95.00	95.00	93.77	99.77	99.00	97.61
CRY	72.89	90.56	87.44	85.33	84.44	89.67	89.11	85.44	73.78	77.22	79.67	86.00	88.56	89.78	86.78
HMS	63.75	64.49	67.65	72.46	67.22	66.76	66.02	69.32	65.25	66.00	65.73	64.87	69.00	67.21	69.22
SMR	73.47	83.85	73.60	72.27	70.26	67.14	86.91	84.68	69.52	78.25	81.04	84.25	73.08	86.45	76.66
URB	85.22	87.92	88.13	82.59	85.81	75.31	87.24	87.53	81.88	83.22	72.96	87.88	72.96	89.85	89.79
IMM	74.33	70.56	74.44	84.11	77.89	67.33	69.44	77.89	69.89	71.78	69.00	75.00	75.22	76.88	77.55
USE	95.74	95.17	97.16	93.22	90.59	84.87	90.51	91.54	54.48	63.68	69.46	85.58	96.93	96.01	96.41
VEC	80.48	85.56	84.53	83.24	81.04	80.46	90.46	80.85	77.09	76.31	76.56	87.43	83.82	85.41	85.32
ECO	92.90	96.86	98.63	98.63	97.77	94.59	96.04	97.81	89.00	90.72	91.22	93.09	98.95	97.22	98.23
GLA	64.19	76.70	72.30	79.35	69.48	72.97	81.05	78.12	64.12	63.51	67.05	81.17	71.96	84.10	77.32
STA	69.77	71.63	70.84	68.90	71.02	69.52	67.77	70.78	59.74	64.00	63.02	66.73	71.25	73.63	68.37
ION	80.45	93.53	83.63	87.42	89.40	79.88	86.74	84.07	85.78	86.18	84.16	90.99	83.27	86.42	90.02
LYM	83.62	81.46	85.15	75.19	83.57	71.08	77.20	82.76	64.87	71.01	70.39	79.60	85.18	82.91	77.86
PAR	72.73	72.48	77.11	84.75	74.08	71.45	94.63	91.37	87.26	90.33	84.59	93.96	85.63	93.75	89.34
BCW	61.36	66.54	71.43	70.52	62.36	56.87	68.20	72.23	66.43	67.14	67.68	67.12	73.92	73.67	74.51
WQT	64.22	73.29	71.48	68.74	69.25	65.90	74.61	69.47	68.74	62.07	65.23	74.37	71.35	73.48	72.38
PID	65.63	73.11	74.66	74.32	71.24	70.34	70.30	73.52	64.81	67.97	66.97	67.83	74.51	74.11	74.72
W,T,L[1]	0,2,18	2,4,14	4,1,15	5,1,14	3,3,14	0,2,18	3,2,15	2,1,17	0,0,20	0,1,19	0,0,20	3,1,17	4,3,13	-,-,-	-,-,-

[1] The last row shows the number of times each method W,T,L (win,tie,loss) compared with IMMIGRATE (**IGT**) by paired t-test.

Table 3: Summarizes the accuracies on UCI datasets.

Data	ADB	SOD	hIN	IM4	imI	**IGT**
BRC	73.21	80.60	77.48	75.74	75.08	75.22
CUS	90.75	90.75	90.31	87.50	87.57	90.05
MON	98.36	91.85	97.15	75.63	75.72	99.51
WIN	97.46	92.92	98.85	98.15	98.23	99.00
CRY	90.44	86.00	87.89	86.22	85.33	89.78
HMS	65.76	67.38	69.40	66.60	66.70	67.21
SMR	81.36	70.54	83.01	76.38	76.43	86.45
URB	87.88	87.88	88.29	87.40	87.23	89.85
IMM	77.22	72.33	70.22	80.66	79.22	76.88
USE	96.01	96.17	96.45	94.14	94.14	96.01
VEC	90.56	85.51	86.48	81.63	81.56	85.41
ECO	98.00	97.86	98.68	97.45	97.50	97.22
GLA	84.96	75.29	75.00	77.97	78.33	84.10
STA	67.23	71.77	69.15	70.75	70.67	73.63
ION	92.10	90.25	92.58	88.32	88.23	86.42
LYM	84.83	79.33	84.76	83.34	83.32	82.91
PAR	90.49	88.18	89.52	83.22	82.64	93.75
BCW	78.18	65.15	71.80	66.41	65.86	73.67
WQT	77.15	69.84	71.22	70.56	70.59	73.48
PID	73.46	75.70	74.11	72.06	72.19	74.11
W,T,L[1]	7,2,11	3,2,15	6,1,13	2,2,16	2,2,16	-,-,-

[1] The last row shows the number of times each method W,T,L (win,tie,loss) compared with IMMIGRATE (**IGT**) by paired t-test.

IM4E-IMMIGRATE. IMMIGRATE outperforms most methods on the UCI machine learning datasets and gene-expression datasets. Its robustness is clearly demonstrated on synthetic dataset where we knew the ground truth. In conclusion, compared with most RELIEF-based algorithms, IMMIGRATE mainly has three advantages: considers both local and global information; uses interaction terms; is robust and less prone to noise.

There are several points in our work which can be explored more. Firstly, since the margin-based framework is not restricted to binary classification scenarios, multiple class classification can be achieved easily based on our work. Secondly, we set a threshold artificially in IM4E-IMMIGRATE if we want to remove small weights to avoid overfitting. $l1$-regularization has been shown by Ng (2004) to have advantages on finding sparse solutions, which is quite valuable for high-dimensional data. Thus, if $l1$-regularization can be imposed into the current cost function, the performance of IMMIGRATE on high-dimensional data may be improved. The overfitting problem encountered in some cases can be avoided. Thirdly, IMMIGRATE only considers pair-wise interactions between features. Interaction terms among multiple features also play an important role in many cases. Our work on generalizing the margin-based framework provides a basis for exploring new algorithms for detecting interactions among multiple features.

Acknowledgements

This research was finished in Harvard University under the guidance of Prof. Pengyu Hong and Prof. Jun S. Liu. Thanks for the kindly help of Prof. Hong and Prof. Liu during my visit in Harvard.

This research was partly supported by Tsinghua University Initiative Scientific Research Program and Spark Program.

References

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750.

Bei, Y. and Hong, P. (2015). Maximizing margin quality and quantity. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE.

Bien, J., Taylor, J., and Tibshirani, R. (2013). A lasso for hierarchical interactions. *Annals of statistics*, 41(3):1111.

Crammer, K., Gilad-Bachrach, R., Navot, A., and Tishby, N. (2003). Margin analysis of the lvq algorithm. In *Advances in neural information processing systems*, pages 479–486.

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

Frank, A. and Asuncion, A. (2010). Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of information and computer science*, 213:2–2.

Freije, W. A., Castro-Vargas, F. E., Fang, Z., Horvath, S., Cloughesy, T., Liau, L. M., Mischel, P. S., and Nelson, S. F. (2004). Gene expression profiling of gliomas strongly predicts survival. *Cancer research*, 64(18):6503–6510.

Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer.

Fukunaga, K. (2013). *Introduction to statistical pattern recognition*. Elsevier.

Gilad-Bachrach, R., Navot, A., and Tishby, N. (2004). Margin based feature selection-theory and algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 43. ACM.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.

Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier.

Kononenko, I. (1994). Estimating attributes: analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer.

Kuhn, H. W. and Tucker, A. W. (2014). Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer.

Li, Y. and Liu, J. S. (2018). Robust variable and interaction selection for logistic regression and general index models. *Journal of the American Statistical Association*, pages 1–16.

Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

Robnik-Šikonja, M. and Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69.

Singh, D., Febbo, P. G., Ross, K., Jackson, D. G., Manola, J., Ladd, C., Tamayo, P., Renshaw, A. A., D'Amico, A. V., Richie, J. P., et al. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209.

Soentpiet, R. et al. (1999). *Advances in kernel methods: support vector learning*. MIT press.

Sun, Y. and Li, J. (2006). Iterative relief for feature weighting. In *Proceedings of the 23rd international conference on Machine learning*, pages 913–920. ACM.

Sun, Y., Todorovic, S., and Goodison, S. (2010). Local-learning-based feature selection for high-dimensional data analysis. *IEEE transactions* *on pattern analysis and machine intelligence*, 32(9):1610–1626.

Sun, Y. and Wu, D. (2008). A relief based feature extraction algorithm. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 188–195. SIAM.

Tian, E., Zhan, F., Walker, R., Rasmussen, E., Ma, Y., Barlogie, B., and Shaughnessy Jr, J. D. (2003). The role of the wnt-signaling antagonist dkk1 in the development of osteolytic lesions in multiple myeloma. *New England Journal of Medicine*, 349(26):2483–2494.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Urbanowicz, R. J., Meeker, M., La Cava, W., Olson, R. S., and Moore, J. H. (2018). Relief-based feature selection: introduction and review. *Journal of biomedical informatics*.

Van't Veer, L. J., Dai, H., Van De Vijver, M. J., He, Y. D., Hart, A. A., Mao, M., Peterse, H. L., Van Der Kooy, K., Marton, M. J., Witteveen, A. T., et al. (2002). Gene expression profiling predicts clinical outcome of breast cancer. *nature*, 415(6871):530.

Yang, M., Wang, F., and Yang, P. (2008). A novel feature selection algorithm based on hypothesis-margin. *JCP*, 3(12):27–34.

Supplementary Material

Proof of Thm. 3.1

1 Since $\mathbf{W}$ is distance metric matrix, it is symmetric and positive-semidefinite. Eigenvalue decomposition of $\mathbf{W}$ is

$$\begin{aligned}\mathbf{W} &= P\Lambda P^T = P\Lambda^{1/2}\Lambda^{1/2}P^T, \\ &= [\sqrt{\lambda_1}\ p_1, \cdots, \sqrt{\lambda_A}\ p_A][\sqrt{\lambda_1}\ p_1, \cdots, \sqrt{\lambda_A}\ p_A]^T,\end{aligned} \tag{5.16}$$

where P is orthogonal matrix. Thus, $\langle p_i, p_j \rangle = 0$.

Let $\Phi = [\phi_1, \cdots, \phi_A] = [\sqrt{\lambda_1}p_1, \cdots, \sqrt{\lambda_A}p_A]$, where $\langle \phi_i, \phi_j \rangle = 0$ and $\lambda_1 > \lambda_2 > \cdots > \lambda_A$.

2 The constraint $\|\mathbf{W}\|_F^2 = 1$ can be simplified since $\mathbf{W}$ can be decomposed to be some orthogonal vectors.

$$\|\mathbf{W}\|_F^2 = \sum_{i,j} w_{i,j}^2 = \sum_i (\phi_i^T\phi_i)^2 = 1 \tag{5.17}$$

3 Let us rearrange the Eq. 3.11.

$$\begin{aligned}&\sum_{h\in\mathcal{H}_n} \alpha_{n,h}\big|\vec{x}_n - \vec{x}_h\big|^T \mathbf{W}\big|\vec{x}_n - \vec{x}_h\big| \\ &= tr(\mathbf{W}\sum_{h\in\mathcal{H}_n}\alpha_{n,h}\big|\vec{x}_n - \vec{x}_h\big|\big|\vec{x}_n - \vec{x}_h\big|^T), \\ &tr(\mathbf{W}\Sigma_{n,H}) = tr(\Sigma_{n,H}\sum_{i=1}^A \phi_i\phi_i^T) = \sum_{i=1}^A \phi_i^T\Sigma_{n,H}\ \phi_i,\end{aligned} \tag{5.18}$$

Then, Eq. 3.11 can be simplified as follows.

$$\begin{aligned}&C = \sum_{i=1}^A \phi_i^T\Sigma\ \phi_i, \\ &subject\ to\ :\|\mathbf{W}\|_F^2 = \sum_i(\phi_i^T\phi_i)^2 = 1, \langle\phi_i, \phi_j\rangle = 0,\end{aligned} \tag{5.19}$$

where $\Sigma = \sum_{n=1}^N \Sigma_{n,H} - \Sigma_{n,M}$ and $\Sigma_{n,H} = \sum_{h\in\mathcal{H}_n}\alpha_{n,h}\big|\vec{x}_n - \vec{x}_h\big|\big|\vec{x}_n - \vec{x}_h\big|^T$, $\Sigma_{n,M} = \beta_{n,m}\big|\vec{x}_n - \vec{x}_m\big|\big|\vec{x}_n - \vec{x}_m\big|^T$.

The orthogonal condition will be ignored by us when deriving the closed form solution because as we can notice at the last step, this condition has already been satisfied.

4 The Lagrangian of Eq. 5.19 is easy to obtain.

$$L = \sum_{i=1}^A \phi_i^T\Sigma\ \phi_i + \lambda(\sum_{i=1}^A(\phi_i^T\phi_i)^2 - 1), \tag{5.20}$$

Derive L with respect to ϕ_i,

$$\partial L/\partial\phi_i = 2\Sigma\phi_i + 4\lambda\phi_i^T\phi_i\phi_i = 0, \tag{5.21}$$

Denote $\phi_i/\|\phi_i\|_2 := \psi_i$. From Eq. 5.21,

$$\Sigma\ \psi_i = \mu_i\ \psi_i, \tag{5.22}$$

where $\mu_i = -2\lambda\|\phi_i\|_2^2$. ψ_i and μ_i are the eigenvector and eigenvalue of Σ separately.

5 Let $\phi_i = \sqrt{\eta_i}\psi_i$, $\eta_i \geq 0$. Thus, $C = \sum_{i=1}^A \sqrt{\eta_i}\psi_i^T\Sigma\sqrt{\eta_i}\psi_i = \sum_{i=1}^A \eta_i\mu_i\psi_i^T\psi_i = \sum_{i=1}^A \eta_i\mu_i$, and $\|\mathbf{W}\|_F^2 = \sum_i(\sqrt{\eta_i}\psi_i^T\sqrt{\eta_i}\psi_i)^2 = \sum_i(\eta_i)^2 = 1$,

Then, Eq. 5.19 can be simplified to be

$$C = \sum_{i=1}^A \eta_i\mu_i,\ subject\ to\ \sum_{i=1}^A(\eta_i)^2 = 1, \eta_i \geq 0 \tag{5.23}$$

6 It is excited to notice Eq. 5.23 is exactly the same as the original Relief Algorithm Eq. 2.1. The solution is

$$\vec{\eta} = (-\vec{\mu})^+/\|(-\vec{\mu})^+\|_2, \tag{5.24}$$

where $(\vec{a})^+ = [max(a_1, 0), max(a_2, 0), \cdots, max(a_I, 0)]$ and $\phi_i = \sqrt{\eta_i}\psi_i$.

Using $\Phi = [\phi_1, \cdots, \phi_A] = [\sqrt{\lambda_1}p_1, \cdots, \sqrt{\lambda_A}p_A]$,

$$\mathbf{W} = \Phi\ \Phi^T \tag{5.25}$$

The orthogonal condition is achieved here because $\|\mathbf{W}\|_F^2 = \sum_i(\phi_i^T\phi_i)^2 = 1$.