MAXIMIZING MARGIN QUALITY AND QUANTITY

Yuanzhe Bei, Pengyu Hong

Brandeis University, Computer Science Department
415 South St. Waltham, MA 02453, USA
{beiyz, hongpeng}@brandeis.edu

ABSTRACT

The large-margin principle has been widely applied to learn classifiers with good generalization power. While tremendous efforts have been devoted to maximizing margins (i.e., the quantity), little attention was paid to ensure the quality of margins. In this paper, we present a new framework that aims to achieve superior generalizability by balancing margin-quantity maximization and margin-quality maximization. In particular, we use one type of margin defined locally by nearest neighbors, and proposes a max-min entropy principle to maximize margin-quality. An iterative algorithm is derived to implement this idea. We demonstrate the power of our new approach by comparing it to a couple of widely used classifiers (e.g., Support Vector Machines, decision tree, naive Bayes classifier, k-nearest neighbors, etc.) and several other large local margin learners (e.g., RELIEF, Simba, G-flip, LOGO, etc.) on a number of UCI machine learning datasets and gene expression datasets.

Index Terms — margin quality, large-margin learning

1. INTRODUCTION

Margins play important roles in learning classifiers with high confidence. In this work, we are interested in hypothesis-margin, which was first implicitly used by the well-known RELIEF algorithm [1] and then was formally defined in [2, 3] in the context of learning feature weights for the purpose of classification. RELIEF calls two different samples as hits of each other if their class labels are the same; otherwise, they are misses of each other. Treating each training sample as a hypothesis, the hypothesis-margin of a training sample can be calculated locally as one half of the difference between its distance to its nearest miss and its distance to its nearest hit. Usually, the larger a margin, the better. Recently several highly competitive learners have been developed within the large hypothesis-margin framework [2-5]. Interestingly, these works all utilized one of the simplest pattern classification methods – the *k* nearest neighbor (*k*-NN) technique [6]. The *k*-NN technique assigns a sample to the class most common among its *k* nearest neighbors. In such a way, it can decompose a globally complex nonlinear separation problem into a set of local and simple ones. In some applications, such as biological and biomedical domains, this can greatly help end-users gain novel insights into a new sample by examining other knowledge associated with its NNs, but is not captured by the features used in classification. In the rest of the paper, we will use margin to denote hypothesis-margin without explicit indication.

A major drawback of RELIEF is that its iterative weight learning procedure does not recalculate the distances between samples after updating feature weights. Hence it can fail to use correct nearest hits/misses in calculating margins during learning. This problem was tackled by G-flip and Simba [3] by re-identifying nearest hits/misses upon changing feature weights. However, RELIEF, G-flip, and Simba all rely on 1-NN, which can make their margin calculations prone to noise and variations in sampling. RELIEF-F [7, 8] tried to remedy this shortcoming by searching for multiple nearest neighbors when computing margins, however, it did not provide a theoretic foundation for deciding the right number of nearest neighbors. In addition, similar to RELIEF, G-flip, and Simba, RELIEF-F chose the nearest hits and misses in deterministic manners, which could significantly undermine their performances. To address this issue, I-RELIEF [4, 9] identified the nearest hits and misses in a probabilistic manner. LOGO [5] improved I-RELIEF by wrapping logistic regression over the expected margins to make the loss function more suitable for classification and applying $\ell 1$-norm regularization on the weight vector to achieve sparseness in selecting features.

Nevertheless, similar to many conventional large-margin approaches, the above methods did not fully appreciate the importance of margin quality. A large margin alone will not lead to good generalization if it is of low quality (e.g., it will be reduced significantly upon losing/corrupting only one or a few critical samples). This problem motivated us to develop a new framework that maximizes margin-quality in addition to maximizing margin-quantity. We incorporated the principle of max-min entropy into our cost function to ensure the quality of margins. As a simple start, we used the weighted Manhattan distance metric. An iterative algorithm was derived to implement this framework. Our new method outperformed several existing methods in the empirical comparisons using UCI machine learning datasets and DNA microarray datasets.

2. THE FRAMEWORK

In a classification problem, we are given a training dataset $\{z_n = (\vec{x}_n, y_n)\}_{n=1\cdots N}$ where $\vec{x}_n \in \Re^D$ is the feature vector and y_n is the class label. To balance margin-quantity maximization and margin-quality maximization, we design a general cost function for learning a classifier with parameters $\vec{w}$ as the following:

$$C = \sum_{n=1}^{N} L_{\vec{w}}(z_n) + \sigma \sum_{n=1}^{N} Q_{\vec{w}}(z_n) + \lambda R(\vec{w}) \qquad (1)$$

where $L_{\vec{w}}(z_n)$ is the loss based on the margin-quantity of the prediction made by the classifier on z_n; and $Q_{\vec{w}}(z_n)$ is the loss based on the quality of z_n's margin. We require that $L_{\vec{w}}(z_n)$ and $Q_{\vec{w}}(z_n)$ monotonically decrease *w.r.t.* the margin-quantity and margin-quality of the prediction made on z_n, respectively. The 1st and 2nd terms of C follow the principle of maximizing margin-quantity and maximizing margin-quality, respectively. The third term of C regularizes $\vec{w}$ to, for example, penalize complex models.

Common regularizations include $\ell 1$-norm, $\ell 2$-norm, and some linear combinations of them. Other constraints on $\vec{w}$ (e.g., non-negative, etc.) can be also imposed. Two positive constants λ and σ are the trade-off parameters that can either be set by users or be tuned by cross validation. There are many potential ways to implement the general framework defined in eq. (1). Below we demonstrate a specific instantiation of this framework.

2.1. Margin-quantity based loss

Following the definitions in [1], z_a is a ***hit*** of z_n if $y_a = y_n$, otherwise z_a is a ***miss*** of z_n. Let $\mathcal{H}_n = \{z_h\}_{y_h=y_n; h\neq n}$ and $\mathcal{M}_n = \{z_m\}_{y_m\neq y_n; m\neq n}$ denote the hit and miss sets of z_n, respectively. For notational simplicity, we use $h \in \mathcal{H}_n$ to indicate $z_h \in \mathcal{H}_n$ and use $m \in \mathcal{M}_n$ to indicate $z_m \in \mathcal{M}_n$.

Margin: Let $z_m \in M_n$ and $z_h \in H_n$ are the nearest miss and hit of z_n, respectively. We adopt the margin definition in [3] as

$$\rho_{n,h,m} = \rho(z_n, z_h, z_m) = f(\vec{x}_n, \vec{x}_m) - f(\vec{x}_n, \vec{x}_h) \quad (2)$$

where $f(\vec{u}, \vec{v})$ measures the distance between $\vec{u}$ and $\vec{v}$. In this work, we use the weighted Manhattan distance $f(\vec{u}, \vec{v}) = \vec{w}^T|\vec{u} - \vec{v}|$, where $\vec{w} \geq 0$ is the feature weight vector to be learned from the training dataset.

Hit/Miss probability variables: If the parameters of $f(\vec{u}, \vec{v})$ are unknown, we are not able to calculate margins defined by eq. (2) because the nearest misses and hits cannot be identified due to the undetermined distances between samples. To deal with this problem, we adopt the method proposed in [4, 5, 9] and use a hidden random variable $\alpha_{n,h}$ to indicate the probability of z_h being the closest hit of z_n, and another hidden random variable $\beta_{n,m}$ to indicate the probability of z_m being the closest miss of z_n. The hit and miss probability variables of z_n should satisfy the constraints $\sum_{h\in\mathcal{H}_n} \alpha_{n,h} = 1$ and $\sum_{m\in\mathcal{M}_n} \beta_{n,m} = 1$. We will show the details for calculating $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$ in Section 3.1.

Margin-quantity based loss: Assuming the training samples are independent, the probability of z_n having a margin $\rho_{n,h,m}$ is $\alpha_{n,h}\beta_{n,m}$. The ***expected margin*** of z_n is equal to $\bar{\rho}_n = \sum_{h\in\mathcal{H}_n}\sum_{m\in\mathcal{M}_n} \alpha_{n,h}\beta_{n,m}\rho_{n,h,m}$, which is used to indicate the ***margin-quantity*** of z_n in this work. We then define the margin-quantity based loss $L_{\vec{w}}(z_n) = -\bar{\rho}_n$.

2.2. Margin-quality and the max-min entropy principle

The margin of a sample, say z_n, will be robust if z_n has many hits in its neighborhood because losing one or a few nearest hits will not affect its margin significantly. In such a scenario, the hit probability distribution of z_n will spread out among many of its hit probability variables, which will lead to a high ***hit-entropy*** $E_{hit}(z_n) = -\sum_{h\in\mathcal{H}_n} \alpha_{n,h} \log \alpha_{n,h}$. On the contrary, if z_n has very few hits in its neighborhood, its hit probability distribution will concentrate at just a few hit probability variables and lead to a low $E_{hit}(z_n)$. In addition, the margin-quality of z_n will not be high, i.e., very likely to decrease upon losing a single nearest hit.

When z_n has very few misses in its neighborhood, its margin quality will also be high because losing one or more nearest misses will most likely increase its margin. In such a scenario, its ***miss-entropy*** $E_{miss}(z_n) = -\sum_{m\in\mathcal{M}_n} \beta_{n,m} \log \beta_{n,m}$ will be low because its miss probability will concentrate at just a few of its miss probability variables. On the contrary, if z_n has many misses in its neighborhood, its miss probability will spread out among many of its miss probability variables, and hence its miss-entropy will be high. In such a scenario, losing one or a few misses will less likely to increase the margin of z_n. Therefore, we can use the hit and miss entropy of a sample to gouge the quality of its margin, and define the ***margin-quality based loss*** as $Q_{\vec{w}}(z_n) = E_{miss}(z_n) - E_{hit}(z_n)$. To minimize $Q_{\vec{w}}(z_n)$, we need to maximize $E_{hit}(z_n)$ and minimize $E_{miss}(z_n)$, which we call ***the max-min entropy principle*** for maximizing margin-quality. A margin with high quality will be less prone to noise, outliers, and sampling variations.

2.3. Instantiate the framework

Here, we derive one instantiation of the framework. Plugging $L_{\vec{w}}(z_n)$ and $Q_{\vec{w}}(z_n)$ defined in Sections 2.1 & 2.2 into the general cost function defined by eq. (1) and applying $\ell 2$-regularization and two constraints on $\vec{w}$, we can easily obtain the following cost function after merging related terms:

$$C = \vec{w}^T \sum_{n=1}^{N} \left(\sum_{h=\mathcal{H}_n} \alpha_{n,h}|\vec{x}_n - \vec{x}_h| - \sum_{m=\mathcal{M}_n} \beta_{n,m}|\vec{x}_n - \vec{x}_m| \right) + \sigma \sum_{n=1}^{N} [E_{miss}(z_n) - E_{hit}(z_n)] + \lambda\|\vec{w}\|^2$$
$$\text{subject to: } \vec{w} \geq 0 \text{ and } \|\vec{w}\|_1 = 1 \quad (3)$$

The none-negative constraint $\vec{w} \geq 0$ is required because we use the Manhattan distance. We limit $\|\vec{w}\|_1 = 1$ because scaling $\vec{w}$ will scale all distances between samples by the same factor. The second term in the right hand side of eq. (3) evaluated margin-quality and is one of our major innovations that differentiates our work from previous large hypothesis-margin methods [2, 3, 5, 9]. Its benefit is explained in Section 2.2 and will be demonstrated in Section 5.1.

3. THE I-M^{4}E ALGORITHM

The cost function defined in eq. (3) is none-convex and complex. To learn an appropriate $\vec{w}$, we derived an iterative algorithm named I-M^{4}E, which stands for **I**terative **M**argin-**M**aximization under **M**ax-**M**in **E**ntropy. I-M^{4}E starts an initial $\vec{w}$, and then iterates between two steps trying to minimize the cost C: (a) fix $\vec{w}$, update $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$; and (b) fix $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$, update $\vec{w}$.

3.1. Fix $\vec{w}$, update $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$

Fixing $\vec{w}$ and setting $\partial C/\partial\alpha_{n,h} = 0$, we can derive the following closed form solution for updating $\alpha_{n,h}$:

$$\frac{\partial C}{\partial \alpha_{n,h}} = \vec{w}^T|\vec{x}_n - \vec{x}_h| + \sigma \log \alpha_{n,h} + \sigma = 0$$
$$\Rightarrow \alpha_{n,h} \propto \exp\left(\frac{-\vec{w}^T|\vec{x}_n - \vec{x}_h|}{\sigma}\right) \quad (4)$$

Normalizing $\alpha_{n,h}$ to ensure $\sum_{h\in H_n} \alpha_{n,h} = 1$, we have

$$\alpha_{n,h} = \exp\left(\frac{-f(\vec{x}_n, \vec{x}_h)}{\sigma}\right) \Big/ \sum_{k\in\mathcal{H}_n} \exp\left(\frac{-f(\vec{x}_n, \vec{x}_k)}{\sigma}\right) \quad (5)$$

Similarly we can derive the following equation for updating $\beta_{n,m}$ while fixing $\vec{w}$

$$\beta_{n,m} = \exp\left(\frac{-f(\vec{x}_n, \vec{x}_m)}{\sigma}\right) \Big/ \sum_{s\in\mathcal{M}_n} \exp\left(\frac{-f(\vec{x}_n, \vec{x}_s)}{\sigma}\right) \quad (6)$$

Interestingly, equations (5) and (6) show that the trade-off constant σ of the margin-quality based loss in the cost function C defines the spread of the exponential kernel (or the effective vicinity range of a sample) in estimating the probabilities $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$. The Hessian matrix of C with respect to any given hit and miss probability pair ($\alpha_{n,h}$, $\beta_{n,m}$) is:

$$\begin{pmatrix} \frac{\partial^2 C}{\partial \alpha_{n,h}{}^2} & \frac{\partial^2 C}{\partial \beta_{n,m} \partial \alpha_{n,h}} \\ \frac{\partial^2 C}{\partial \alpha_{n,h} \partial \beta_{n,m}} & \frac{\partial^2 C}{\partial \beta_{n,m}{}^2} \end{pmatrix} = \begin{pmatrix} \frac{\sigma}{\alpha_{n,h}} & -\rho_{n,h,m} \\ -\rho_{n,h,m} & -\frac{\sigma}{\beta_{n,m}} \end{pmatrix} \quad (7)$$

Since both $\alpha_{n,h}$ and $\beta_{n,m}$ are positive, the determinant of the above matrix is negative:

$$-\frac{\sigma^2}{\alpha_{n,h}\beta_{n,m}} - \rho_{n,h,m}{}^2 < 0 \quad (8)$$

Thus, when $\vec{w}$ is fixed, the I-M^{4}E algorithm finds a saddle point in the $(\alpha_{n,h}, \beta_{n,m})$ space.

3.2. Fix $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$, update $\vec{w}$

When $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$ are fixed, the cost C is quadratic of $\vec{w}$. Moreover, the second derivative of C with respect to $\vec{w}$ is $2\lambda > 0$. Hence, C is convex with respect to $\vec{w}$, and a $\vec{w}$ satisfying $\vec{w} \geq 0$ & $\|\vec{w}\|_1 = 1$ can be found to globally minimize C while fixing $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$. However, such a local minimizer may not necessary achieve a good final result because $\{\alpha_{n,h}\}$ and $\{\beta_{n,m}\}$ depend on $\vec{w}$. In addition, it requires more computational resources. Therefore, we derived the following efficient way for updating $\vec{w}$ to achieve a reasonably good result. Our experiments showed that this method worked very well empirically. Let $\partial C / \partial \vec{w} = 0$, we can obtain a constraint free solution to update $\vec{w}$:

$$\vec{v} = \frac{1}{2\lambda}\left[\sum_{n=1}^{N}\left(\sum_{m \in \mathcal{M}_n} \beta_{n,m}|\vec{x}_n - \vec{x}_m| - \sum_{h \in \mathcal{H}_n} \alpha_{n,h}|\vec{x}_n - \vec{x}_h|\right)\right]$$

Then, factoring in the constraint $\vec{w} \geq 0$, we easily obtain a solution for $\vec{w}$ as $\vec{v}^+ = [\max(v_1, 0) \cdots \max(v_D, 0)]^T$. Since C is convex with respect to $\vec{w}$, $\vec{v}^+$ is a straightforward none-negative solution that globally minimizes C. Adding the constraint $\|\vec{w}\|_1 = 1$, we derived the following sub-optimal but efficient way to update $\vec{w}$:

$$\vec{w} = \vec{v}^+ / \|\vec{v}^+\|_1 \quad (9)$$

We of course can use the result given by eq. (9) as the initial point, and perform simple binary search along the line defined by $\|\vec{w}\|_1 = 1$ to find a $\vec{w}$ that minimizes C.

3.3. The iterative algorithm

The above iterative procedure is summarized in Algorithm 1. Based on the analysis in Section 3.1 and 3.2, this algorithm should converge to a saddle point or a local minimal.

4. CLASSIFY NEW SAMPLES

We design a new rule as the following to classify a new sample $z' = (\vec{x}', y')$ using the learned $\vec{w}$:

$$y' = \text{argmin}_c \, \Psi_c = \text{argmin}_c \sum_{y_n = c} P_n^c f(\vec{x}_n, \vec{x}') \quad (10)$$

where c indicates the class and $P_n^c = \frac{\exp\left(\frac{-f(\vec{x}_n, \vec{x}')}{\sigma}\right)}{\sum_{y_k = c} \exp\left(\frac{-f(\vec{x}_k, \vec{x}')}{\sigma}\right)}$.

Here Ψ_c measures the expected distance of z' to the c-th class. Eq. (10) assigns z' to the class whose expected distance to z' is the minimum among all classes. There is a close connection between Ψ_c and the margin-quantity based loss $L_{\vec{w}}$ of assigning z' to the c-th class (*i.e.*, $y' = c$):

$$L_{\vec{w}}(y' = c) = \Psi_c - \sum_{c' \neq c} \Psi_{c'} \quad (11)$$

Algorithm 1: I-M^{4}E

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 $\{\alpha_{n,h}{}^{(t)}\}$ and $\{\beta_{n,m}{}^{(t)}\}$ using equations (5) and (6), respectively.

Calculate $w^{(t)}$ using equation (9).

Until the change of C is small enough or the number of iteration t reaches a pre-set limit.

Return: $w^{(t)}$

Obviously, $L_{\vec{w}}(y' = c)$ will be the minimum among all possible class assignments to z' if the corresponding Ψ_c is the minimum.

5. EXPERIMENTS

5.1. An illustration

To illustrate the benefits of considering margin-quality, we designed the following two simple examples of binary classification with feature selection in 3-dimensional space. In both examples, we focus on a target sample z_t and compare two feature weight vector options: (1) $\vec{w}_{(1,2)} = [0.5, 0.5, 0]^T$ that selects the 1st & 2nd dimensions *vs.* (2) $\vec{w}_{(2,3)} = [0, 0.5, 0.5]^T$ that selects the 2nd & 3rd dimensions.

The first example (Figure 1 and Table 1) highlights z_t and one of its hits z_r. In the sub-space selected by $\vec{w}_{(1,2)}$, z_t has many nearest hits evenly distributed around it, which leads to a higher hit-entropy = 1.778. Losing z_r does not greatly reduce the expected margin $\bar{\rho}_t$ of z_t, and the margin-quantity based loss $L_{\vec{w}}(z_t)$ is increased slightly from -0.959 to -0.894. In the sub-space selected by $\vec{w}_{(2,3)}$, although z_t has a larger $\bar{\rho}_t$, the nearest hits of z_t are unevenly distributed around it since z_r much closer to z_t than others. This leads to a smaller hit-entropy = 1.585. Losing z_r causes a significant reduction to $\bar{\rho}_t$ and dramatically increases $L_{\vec{w}}(z_t)$ from -1.132 to -0.763. Such a big change, which can be forewarned by examining $E_{hit}(z_t)$, indicates that the margin of z_t in the subspace selected by $\vec{w}_{(2,3)}$ is not robust. If $L_{\vec{w}}(z_t)$ is the only factor, $\vec{w}_{(2,3)}$ will be preferred over $\vec{w}_{(1,2)}$ because $\vec{w}_{(2,3)}$ leads to a smaller loss (-1.132 *vs.* -0.959). If both $L_{\vec{w}}(z_t)$ and $Q_{\vec{w}}(z_t)$ are considered, $\vec{w}_{(1,2)}$ will be a better option because it makes the margin of z_t more robust and leads to a smaller total-loss [$L_{\vec{w}}(z_t)$ + $Q_{\vec{w}}(z_t)$] with/without z_r.

The second example (Figure 2 and Table 2) highlights z_t and one of its misses z_q. In the sub-space selected by $\vec{w}_{(1,2)}$, z_t has many nearest misses so that z_t has a higher miss-entropy $E_{miss}(z_t)$=1.933. Losing z_q does not greatly affect $\bar{\rho}_t$, and only reduces $L_{\vec{w}}(z_t)$ slightly from -0.514 to -0.515. On the contrary, in the sub-space selected by $\vec{w}_{(2,3)}$, the expected margin $\bar{\rho}_t$ is mainly decided by z_q because z_q is much closer to z_t than the rest nearest misses, which makes z_q more like an outlier. Such an uneven distribution of the nearest misses around z_t leads to a relatively smaller miss-entropy $E_{miss}(z_t)$=1.863. Losing z_q will significantly increase $\bar{\rho}_t$, and hence dramatically reduce $L_{\vec{w}}(z_t)$ from -0.464 to -0.667. Such a big change, which can be predicted by examining $E_{miss}(z_t)$, indicates that the margin of z_t in the sub-space defined by $\vec{w}_{(2,3)}$ is of higher quality because it will be significantly better upon losing any single outlier-like miss. In other words, $Q_{\vec{w}}(z_t)$

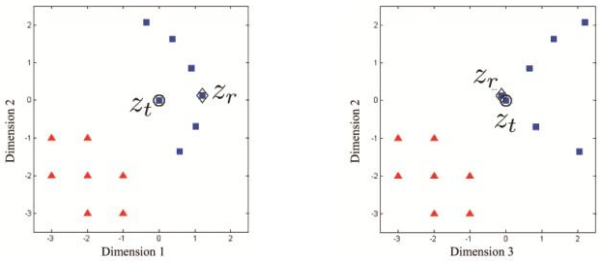

Figure 1: The benefit of incorporating hit entropy into features selection. There are two classes (▲ *vs.* ■) in 3D space. The left and the right plots show the subspaces selected by $\vec{w}_{(1,2)}$ and $\vec{w}_{(2,3)}$, respectively.

	Before losing z_r				After losing z_r		
	$L_{\vec{w}}$	E_{hit}	$Q_{\vec{w}}$	$T_{\vec{w}}$	$L_{\vec{w}}$	$Q_{\vec{w}}$	$T_{\vec{w}}$
$\vec{w}_{(1,2)}$	-0.959	1.778	0.099	-0.860	-0.894	0.276	-0.618
$\vec{w}_{(2,3)}$	-1.132	1.585	0.292	-0.840	-0.763	0.397	-0.366

Table 1: Compares $L_{\vec{w}}$, $Q_{\vec{w}}$, and $T_{\vec{w}} = L_{\vec{w}} + Q_{\vec{w}}$ of z_t before and after losing z_r in Figure 1. The miss entropy (not shown) remains the same before and after losing z_r.

helps choose a margin more robust to outliers in the opposite class(es). If $L_{\vec{w}}(z_t)$ is the only factor, $\vec{w}_{(1,2)}$ will be better than $\vec{w}_{(2,3)}$. If we also consider $Q_{\vec{w}}(z_t)$, $\vec{w}_{(2,3)}$ will be a better choice because it leads to a smaller total-loss with/without z_q.

5.2. Results of real datasets

We carried out cross validation experiments to compare I-M^{4}E with several popular classifiers implemented in WEKA [10], such as decision tree (ADT) [11], naive Bayes classifier (NBC) [12], Bayesian network (BN) [13, 14], Support Vector Machines trained by sequential minimal optimization (SMO) [15, 16], RBF network [17] and 1-nearest neighbor (IB1) and 3-nearest neighbor (IB3) [18]. We also compared our method to several other margin-based nearest neighbor algorithms, such as Simba (SIM) [3], G-flip (GFP) [3], RELIEF (RLF) [1], RELIEF-F (RFF) [7, 8], and LOGO [5]. Simba, G-flip and RELIEF were implemented in a MATLAB package tool generously made available by Navot [19]. RELIEF-F is available in the Statistics Toolbox of MATLAB. The MATLAB codes of LOGO were generously provided by one of its inventors. The settings of these methods were chosen according to those suggested in their original papers: RELIEF uses 1-NN classifier and the Euclidean distance; Simba and G-flip use 1-NN classifier and the weighted Euclidean distance; LOGO uses 3-NN classifier and the weighted Manhattan distance, and the value of its kernel width is decided from a set of values by internal cross validation; and RELIEF-F uses the Manhattan distance and a *k*-NN classifier (the value of k = 1, 3, or 5 was decided using internal cross validation). The I-M^{4}E algorithm has two constants λ and σ in its cost function. We fixed λ as the number of samples and tuned σ by internal cross validation in the following way: start from $\sigma_0 = 1.0$, and gradually reduce σ by half each time (i.e., $\sigma_i = \sigma_0/2^i$) until the value of σ_i is smaller than 0.0001. The best σ_k was chosen as the one performed the best in the internal cross validation, and then was used in the corresponding test. The same strategy was used to choose the best kernel width σ for LOGO. We also tried eq. (10) to classify new samples using $\vec{w}$ learned by LOGO. However, the results were in general much worse than those of using the 3-NN classifier suggested in [5].

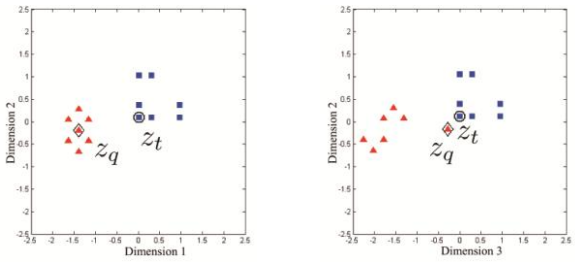

Figure 2: The benefit of incorporating miss entropy into selecting features in a 3D binary classification (▲ *vs.* ■) problem. The left and the right plots show the subspaces selected by $\vec{w}_{(1,2)}$ and $\vec{w}_{(2,3)}$, respectively.

	Before losing z_q				After losing z_q		
	$L_{\vec{w}}$	E_{miss}	$Q_{\vec{w}}$	$T_{\vec{w}}$	$L_{\vec{w}}$	$Q_{\vec{w}}$	$T_{\vec{w}}$
$\vec{w}_{(1,2)}$	-0.514	1.933	0.164	-0.350	-0.515	0.008	-0.507
$\vec{w}_{(2,3)}$	-0.464	1.863	0.095	-0.369	-0.667	-0.016	-0.683

Table 2: Compares $L_{\vec{w}}$, $Q_{\vec{w}}$, and $T_{\vec{w}} = L_{\vec{w}} + Q_{\vec{w}}$ of z_t before and after losing z_q in Figure 2. The hit entropy (not shown) remains the same before and after losing z_q.

In each round of 10-fold cross validation, the accuracy of each method was measured as the percentage of correctly predicted test samples. The 10-fold cross validation process was run 10 times for each datasets so that each algorithm generated $10 \times 10 = 100$ cross validation results. To decide if a classifier performed significantly better than the other one on a specific dataset, we adopted the method proposed in [5] and used Student's paired two-tailed *t*-test [20] to compare the accuracies of two different classifiers in all cross validation runs. We say a classifier performs noticeably better than (i.e., *win*) another one if the paired *t*-test $\boldsymbol{p < 0.01}$ and the average accuracy of the first classifier is at least **0.5%** higher than that of the second one. Otherwise, we decide that two classifiers are comparable or tie.

5.2.1. Results of clinical gene expression datasets

We carried out the comparisons using five publicly available clinical gene expression datasets: Breast (44 samples with disease-free ≥ 5 yrs *vs.* 34 samples < 5 yrs) [22], Colon (40 tumor tissue samples *vs.* 22 normal tissue samples) [23], GLI (26 grade III samples *vs.* 59 grade IV samples) [24], Myeloma (36 samples with lesions not detected by MRI *vs.* 137 samples detected by MRI) [25], and Prostate (59 normal tissue samples *vs.* 77 tumor tissue samples) [26]. The number of features in each of these gene expression datasets is much larger than its number of samples. It is widely known that feature selection will help obtain better classification results in high dimensional spaces. Similar to RELIEF, Simba, G-flip, RELIEF-F, and LOGO which were originally derived for the purpose of feature selection, I-M^{4}E can rank features by their weights. Decision tree learning can automatically identify good features in building tree classifiers. To select features for SMO, we used SVM Recursive Feature Elimination (SVM-RFE) [27] which is a popular SVM-based feature ranking algorithm. Several kernels were tested for SMO, such as Poly kernel with exponent 1 (SVM1), 2 (SVM2), and 3 (SVM3), RBF (SVMR) and PUK (SVMP).

The following feature elimination scheme was used for all of them. Starting from using all features *N*, we removed 50% of the features with the lowest ranks at a time until the number of features is smaller than 50. The best number of features for each algorithm

Dataset	LOGO	RFF	SIM	RLF	ADT	GFP	SVM1	SVM2	SVM3	SVMP	SVMR	I-M4E
Breast	87.21	94.92	92.37	91.46	86.88	92.71	92.33	90.25	90.87	86.88	92.08	94.08
Colon	81.83	83.21	80.17	80.50	76.04	79.67	82.00	80.33	77.00	69.58	81.37	85.50
GLI	84.89	87.72	85.61	84.78	83.69	87.97	90.53	90.64	90.75	76.06	89.44	89.75
Myeloma	93.92	97.79	94.94	96.25	76.52	97.68	79.00	79.23	79.17	79.17	79.17	98.77
Prostate	89.83	88.16	85.84	86.59	86.97	87.39	91.51	92.16	92.30	83.01	69.86	88.13
W/T/L	1/0/4	1/1/3	0/0/5	0/0/5	0/0/5	0/0/5	2/0/3	2/0/3	2/0/3	0/0/5	0/1/4	-/-/-

Table 3: Summarizes the results on five high-dimensional gene expression datasets. The first column lists the datasets, and the first row lists the methods in comparison. The last row indicates the number of times each algorithm W/T/L (***win/tie/loss***) when compared to I-M^{4}E using Student's paired two-tailed *t*-test. The remaining rows list the average accuracies of these algorithms on the corresponding datasets.

Dataset	BN	NBC	IB1	IB3	RBF	I-M4E
Breast	85.38	77.88	86.96	86.13	85.21	92.04
Colon	78.25	62.63	71.54	77.88	72.63	82.71
GLI	84.86	82.42	89.75	89.08	84.61	90.81
Myeloma	74.09	60.30	90.03	92.97	70.65	97.38
Prostate	65.53	56.20	84.11	87.86	55.71	92.26
W/T/L	0/0/5	0/0/5	0/0/5	0/0/5	0/0/5	-/-/-

Table 4: Compares I-M^{4}E with BN, NBC, IB1, IB3, and RBF using five high-dimensional gene expression datasets. The first column lists the datasets, and the first row lists the algorithms in comparison. The Chi-square attribute selector in Weka is used to select the best 2000 features during training. The last row indicates the number of times each algorithm W/T/L(i.e., ***win/tie/loss***) when compared to I-M^{4}E using the Student's paired two-tailed *t*-test. The remaining rows list the average accuracies of those algorithms. I-M^{4}E is a clear winner in this comparison.

was decided using internal 10-fold cross validation. Then each classifier was retrained using all training samples and the features selected in the internal cross validation before it was applied to the test samples. The comparison results (Table 3) show that our method significantly outperformed Simba, RELIEF and SVM (PUK kernel) on all five datasets, and also significantly outperformed LOGO, G-flip, SVM (RBF kernel) on four out of five datasets. SVM (Poly kernel with exponent 1, 2 and 3) beats I-M^{4}E on the GLI and Prostate datasets by (0.78%, 0.89% and 1%) and (3.38%, 4.03% and 4.17%), respectively. However, I-M^{4}E has a much larger lead over them on Myeloma (by 19.77%, 19.54% and 19.6%) datasets and a larger lead on Colon (by 3.5%, 5.17% and 8.33%), and a reasonable lead on the Breast dataset (by 1.75%, 3.83% and 3.21%).

Other classifiers (such as BN, NBC, IB1, IB3, and RBF network) do not have their own intrinsic feature ranking/selection functions. In each 10-fold cross validation run, we first applied the Chi-square attribute selector in WEKA to select the best 2000 features using internal 10-fold cross validation, and then trained I-M^{4}E and these classifiers for comparison on the test subset. The results clearly indicate that I-M^{4}E significantly outperformed these classifiers (see Table 4).

5.2.2. Results of UCI datasets

Eleven UCI datasets [21]: Vertebral column (VC), Ecoli (EC), Glass identification (GI), Haberman's survival (HS), Hayes-roth (HR), Statlog/heart (SH), Ionosphere (IONO), Lymphography (LYM), Parkinsons (PARK), Wine Quality (WQ), and Wisconsin Prognostic Breast Cancer (WPBC) were used in this experiment. Some of these UCI datasets have multiple classes. We only used the largest two classes in each one of them. All features were standardized to center at 0 with unit variance. We tested several kernels (such as PolyKernel with exponent 1, 2, and 3, RBF, and PUK with default settings in WEKA) for SMO and used the best performance in comparison. The results are summarized in Table 5 and show that I-M^{4}E in general outperforms other methods though it is not always the best.

6. CONCLUSIONS AND DISCUSSIONS

In this paper, we present a new learning framework and a simple algorithm I-M^{4}E to implement it. The convergence analysis of I-M^{4}E is straightforward. The empirical results showed that I-M^{4}E is promising. The instantiation and implementation of the framework can be improved in several fronts. Currently, we use a simple linear margin-based loss function (i.e., the negation of a margin) which may limit its performance in classification tasks. Nevertheless, our formulation is general enough to easily adopt other margin-based loss functions (e.g., logistic loss, hinge loss, sigmoidal loss, etc.) to make the cost function much more appropriate for the purpose of classification. The same iterative algorithm can be used with some modifications. The search of the weight vector may have to resort to gradient-based methods because a closed form solution may not exist for a non-linear loss function. We now impose $\ell 2$-regularization on the feature weight vector. $\ell 2$-regularization has been shown by several works to be inferior to $\ell 1$-regularization [28] in finding sparse solutions [5, 29], which is desirable in some applications. Our future work will investigate using $\ell 1$-regularization to implement this framework. So far, we have demonstrated the framework on binary classification problems. Since the definitions of the hit/miss sets, margin, and cost can be readily applied to multiple class scenarios, we do not foresee any theoretic barrier to apply this work to multiple class problems.

Our work is related to distance metric learning [30-35] in the sense that I-M^{4}E learns a weighted Manhattan distance metric. Learning other distance metrics (e.g., Mahalanobis distance and weighted Euclidean distance) can be formulated within our framework. The equations for calculating the parameters of these distance metrics should be derived correspondingly. Our approach was inspired by previous large hypothesis-margin methods [2, 3, 5, 9], especially I-RELIEF [9] and LOGO [5]. One of our major innovations is that our objective function evaluates margin-quality, which was not considered by previous large hypothesis-margin methods. In the instantiation of the framework presented here, we deployed the maximum hit-entropy and minimal miss-entropy principle to make margin calculation less prone to noise and sampling variations, and showed that this principle could have great impact on the results (Figures 1 & 2). This was one of the major factors that contributed to our better performances in experiments. Both I-RELIEF and LOGO also use equations (5) & (6) to calculate the hit and miss probabilities, respectively. However, their choices of these two equations were made intuitively and preceded the designs of their objective functions. Since the hit and miss probabilities are not independent from the

Dataset	ADT	BN	IB1	IB3	NBC	RBF	SMO	SIM	GFP	RLF	RFF	LOGO	I-M4E
VC	82.43	76.60	80.85	77.78	77.79	80.31	78.99	81.08	82.02	84.50	79.64	83.65	82.93
EC	98.82	98.67	97.22	98.45	97.70	98.32	97.82	97.57	97.39	97.21	97.82	98.91	98.40
GI	92.52	92.18	94.57	94.86	90.59	92.94	92.63	94.46	93.87	93.95	93.41	93.98	94.39
HS	72.91	71.72	65.65	70.13	74.59	73.93	73.11	64.67	65.81	63.99	69.03	68.74	72.32
HR	75.35	38.52	75.14	64.42	67.39	59.69	60.52	79.31	52.51	60.84	76.24	77.10	79.59
SH	79.94	82.76	75.52	78.91	83.97	82.99	81.24	75.33	71.93	74.72	81.46	79.22	83.28
IONO	90.23	89.52	86.75	85.70	82.52	91.45	90.74	89.33	88.34	88.62	90.80	91.82	92.09
LYM	82.70	81.49	75.65	81.19	84.09	83.37	84.54	77.60	65.69	79.81	85.75	85.26	84.94
PARK	88.11	79.59	95.88	93.44	69.92	80.99	89.16	95.69	94.32	93.74	92.79	94.27	94.18
WQ	70.22	66.29	76.30	70.26	66.28	66.58	69.85	76.99	76.55	77.10	77.77	53.99	78.44
WPBC	75.39	75.79	71.18	74.43	67.18	76.15	77.84	69.63	64.43	67.30	76.02	74.96	76.25
W/T/L	1/1/9	0/2/9	1/1/9	0/2/9	2/0/9	1/3/7	2/1/8	1/2/8	0/1/10	1/2/8	1/1/9	2/4/5	-/-/-

Table 5: Summarizes the results on eleven UCI datasets. The first column lists the datasets, and the first row lists the methods. The last row indicates the number of times each algorithm W/T/L (***w**in*/***t**ie*/***l**oss*) when compared to I-M^{4}E using Student's paired two-tailed *t*-test. The remaining rows list the average accuracies of these methods on the corresponding datasets.

feature weights, it is not clear how updating the hit and miss probabilities in such ways affects the optimization of their objective functions in terms of finding the feature weights. I-M^{4}E derives equations (5) & (6) by directly optimizing our novel objective function eq. (1) iteratively. The theoretical foundation and consequence of updating the hit and miss probabilities by these two equations in I-M^{4}E are apparent and sound.

7. REFERENCES

[1] Kira, K. and L.A. Rendell, *A practical approach to feature selection, International Workshop on Machine Learning*. 1992: Aberdeen, Scotland, United Kingdom. p. 249-256.

[2] Crammer, K., et al., *Margin Analysis of the LVQ Algorithm.*, in *NIPS*. 2002, MIT Press. p. 462-469.

[3] Gilad-Bachrach, R., A. Navot, and N. Tishby, *Margin based feature selection - theory and algorithms*. ICML 2004, p. 43-50.

[4] Sun, Y. and J. Li, *Iterative RELIEF for feature weighting*. ICML 2006, p. 913-920.

[5] Sun, Y., S. Todorovic, and S. Goodison, *Local-Learning-Based Feature Selection for High-Dimensional Data Analysis.* IEEE Trans. Pattern Anal. Mach. Intell., 2010. **32**(9): p. 1610-1626.

[6] Cover, T. and P. Hart, *Nearest neighbor pattern classification.* IEEE Transactions on Information Theory, 1967. **13**(1): p. 21-27.

[7] Kononenko, I., *Estimating attributes: Analysis and extensions of RELIEF*, in *Proceedings of ECML'94*. 1994. p. 171-182.

[8] Robnik-Šikonja, M. and I. Kononenko, *Theoretical and Empirical Analysis of ReliefF and RReliefF*. Mach. Learn., 2003. **53**(1-2): p. 23-69.

[9] Sun, Y., *Iterative RELIEF for Feature Weighting: Algorithms, Theories, and Applications.* IEEE Trans. Pattern Anal. Mach. Intell., 2007. **29**(6): p. 1035-1051.

[10] Frank, E., et al. *Weka:* http://www.cs.waikato.ac.nz/ml/weka/ .

[11] Freund, Y. and L. Mason, *The Alternating Decision Tree Learning Algorithm*. 1999: San Francisco, CA, USA. p. 124-133.

[12] John, G.H. and P. Langley, *Estimating continuous distributions in Bayesian classifiers*, UAI 1995, p. 338-345.

[13] Heckerman, D., D. Geiger, and D.M. Chickering, *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.* Machine Learning, 1995. **20**(3): p. 197-243.

[14] Friedman, N., D. Geiger, and M. Goldszmidt, *Bayesian Network Classifiers.* Mach. Learn., 1997. **29**(2-3): p. 131-163.

[15] Platt, J.C., *Advances in kernel methods*. 1999, MIT Press: Cambridge, MA, USA. p. 185-208.

[16] Keerthi, S.S., et al., *Improvements to Platt's SMO Algorithm for SVM Classifier Design.* Neural Comp., 2001.**13**(3): p.637-649.

[17] Haykin, S., *Neural Networks: A Comprehensive Foundation*. 1998, Upper Saddle River, NJ, USA: Prentice Hall PTR.

[18] Aha, D.W., D. Kibler, and M.K. Albert, *Instance-Based Learning Algorithms.* Mach. Learn., 1991. **6**(1): p. 37-66.

[19] Navot, A. http://cid-843f36fc50a5ece8.skydrive.live.com/browse.aspx/.Public/Feature%20Selection.

[20] Gosset, W.S., *The Probable Error of a Mean.* Biometrika, 1908. **6**(1): p. 1-25.

[21] Frank, A. and A. Asuncion. *UCI Machine Learning Repository*. 2010.

[22] van 't Veer, L.J., et al., *Gene expression profiling predicts clinical outcome of breast cancer.* Nature, 2002. **415**(6871): p. 530-536.

[23] Alon, U., et al., *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays.* PNAS, 1999. **96**(12): p. 6745-6750.

[24] Freije, W.A., et al., *Gene expression profiling of gliomas strongly predicts survival.* Cancer Res, 2004. **64**(18): p. 6503-10.

[25] Tian, E., et al., *The role of the Wnt-signaling antagonist DKK1 in the development of osteolytic lesions in multiple myeloma.* N Engl J Med, 2003. **349**(26): p. 2483-94.

[26] Singh, D., et al., *Gene expression correlates of clinical prostate cancer behavior.* Cancer Cell, 2002. **1**(2): p. 203-209.

[27] Guyon, I., et al., *Gene Selection for Cancer Classification using Support Vector Machines.* Mach. Learn., 2002. **46**(1-3): p. 389-422.

[28] Donoho, D.L. and M. Elad, *Optimally sparse representation in general (nonorthogonal) dictionaries via l-minimization.* PNAS, 2003. **100**(5): p. 2197-2202.

[29] Ng, A.Y., *Feature selection, L1 vs. L2 regularization, and rotational invariance*. ICML 2004, p. 78-85.

[30] Hastie, T. and R. Tibshirani, *Discriminant Adaptive Nearest Neighbor Classification.* IEEE Trans. PAMI 1996. **18**(6): p. 607-616.

[31] Xing, E.P., et al., *Distance Metric Learning, with Application to Clustering with Side-information*, in *NIPS*. 2002,

[32] Goldberger, J., et al., *Neighborhood components analysis*, in *NIPS*. 2004. p. 513-520.

[33] Davis, J.V., et al., *Information-theoretic metric learning*. 2007, ACM Press: Corvalis, Oregon, USA. p. 209-216.

[34] Weinberger, K.Q. and L.K. Saul, *Distance Metric Learning for Large Margin Nearest Neighbor Classification.* J. Mach. Learn. Res., 2009. **10**: p. 207-244.

[35] Ying, Y., K. Huang, and C. Campbell, *Sparse Metric Learning via Smooth Optimization*, in *NIPS*. 2009: Vancouver, British Columbia, Canada. p. 2214-2222.