Multi-label Classification Using Error-correcting Codes of Hard or Soft Bits

Chun-Sung Ferng, and Hsuan-Tien Lin, *Member, IEEE*

***Abstract*—We formulate a framework for applying error-correcting codes (ECC) on multi-label classification problems. The framework treats some base learners as noisy channels and uses ECC to correct the prediction errors made by the learners. An immediate use of the framework is a novel ECC-based explanation of the popular random k-label-sets (RAKEL) algorithm using a simple repetition ECC. Using the framework, we empirically compare a broad spectrum of off-the-shelf ECC designs for multi-label classification. The results not only demonstrate that RAKEL can be improved by applying some stronger ECC, but also show that the traditional Binary Relevance approach can be enhanced by learning more parity-checking labels. Our study on different ECC also helps understand the trade-off between the strength of ECC and the hardness of the base learning tasks. Furthermore, we extend our study to ECC with either hard (binary) or soft (real-valued) bits by designing a novel decoder. We demonstrate that the decoder improves the performance of our framework.**

***Index Terms*—Multi-label classification, error-correcting codes.**

I. Introduction

MULTI-LABEL classification is an extension of traditional multi-class classification. In particular, the latter aims at accurately associating one single label with an instance, while the former aims at associating a label set. Because of the increasing application needs in domains like music categorization [1] and scene analysis [2], multi-label classification is attracting much research attention in recent years.

Error-correcting code (ECC) roots from the information theoretic pursuit of communication [3]. In particular, the ECC studies how to accurately recover a desired signal block after transmitting the block's encoding through a noisy communication channel. When the desired signal block is the single label (of some instances) and the noisy channel consists of some binary classifiers, it has been shown that a suitable use of the ECC could improve the association (prediction) accuracy of multi-class classification [4]. Several designs, including some classic ECC [4] and some adaptively constructed ECC [5], [6], have reached promising empirical performance for multi-class classification.

While the benefits of the ECC are well established for multi-class classification, the corresponding use for multi-label classification remains an ongoing research direction. [7] takes the first step in this direction by proposing a multi-label classification approach that applies a classic ECC, the Bose-Chaudhuri-Hocquenghem (BCH) code. The work is followed by some extensions to the convolution code [8]. Although the approach shows some good experimental results over existing multi-label classification approaches, a more rigorous study remains needed to understand the advantages and disadvantages of different ECC designs for multi-label classification and will be the main focus of this work.

In this work, we formalize the framework for applying the ECC on multi-label classification. The framework is more general than both existing ECC studies for multi-class classification [4] and for multi-label classification [7]. Then, we conduct a thorough study with a broad spectrum of classic ECC designs: repetition code, Hamming code, BCH code, and low-density parity-check code. The four designs cover the simplest ECC idea to the state-of-the-art ECC in communication systems. Interestingly, such a framework allows us to give a novel ECC-based explanation to the random k-label sets (RAKEL) algorithm, which is popular for multi-label classification. In particular, RAKEL can be viewed as a special type of repetition code coupled with a batch of simple and internal multi-label classifiers.

We empirically demonstrate that RAKEL can be improved by replacing its repetition code with the Hamming code, a slightly stronger ECC. Furthermore, even better performance can be achieved when replacing the repetition code with the BCH code. When compared with the traditional Binary Relevance (BR) approach without the ECC, multi-label classification with the ECC can perform significantly better. The empirical results justify the validity of the ECC framework.

In addition, we design a new decoder for linear ECC by using multiplications to approximate exclusive-OR operations. This decoder is able to handle not only ordinary binary bits from the channels, called *hard inputs*, but also real-valued bits, called *soft inputs*. For multi-label classification using the ECC, the soft inputs can be used to represent the confidence of the internal classifiers. Our newly designed decoder allows a proper use of the detailed confidence information to produce more accurate predictions. The experimental results show that this decoder indeed improves the performance of the ECC framework with soft inputs.

The paper is organized as follows. First, we introduce the multi-label classification problem in Section I-A, and present related works in Section I-B. Section II illustrates the framework and demonstrates its effectiveness. Section III presents a new decoder for hard or soft inputs. We empirically study the performance of the framework and the proposed decoder in Section IV. Finally we conclude in Section V.

C.-S. Ferng and H.-T. Lin are with the Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, e-mail: {r99922054, htlin}@csie.ntu.edu.tw.

Manuscript received August ??, 2012; revised ??.

A short version of the paper appeared in the 2011 Asian Conference on Machine Learning [9]. The paper was then enriched by the novel decoder for dealing with soft bits, the comparison with other ECC designs, and broader experiments. The paper is also the core of the first author's M.S. thesis [10].

A. Problem Setup

Multi-label classification aims at mapping an instance $\mathbf{x} \in \mathbb{R}^d$ to a label-set $Y \subseteq \mathcal{L} = \{1, 2, \ldots, K\}$, where K is the number of classes. Following the hypercube view of [11], the label set Y can be represented as a binary vector $\mathbf{y}$ of length K, where $\mathbf{y}[i]$ is 1 if the ith label is in Y, and 0 otherwise. Consider a training dataset $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$. A multi-label classification algorithm uses D to locate a multi-label classifier $h\colon \mathbb{R}^d \to \{0,1\}^K$ such that $h(\mathbf{x})$ predicts $\mathbf{y}$ well on future test examples $(\mathbf{x}, \mathbf{y})$.

There are several loss functions for evaluating whether $\tilde{\mathbf{y}} = h(\mathbf{x})$ predicts $\mathbf{y}$ well. Two common ones are:

- **subset** $0/1$ **loss**: $\Delta_{0/1}(\tilde{\mathbf{y}}, \mathbf{y}) = [\![\tilde{\mathbf{y}} \neq \mathbf{y}]\!]$.
- **Hamming loss**: $\Delta_{HL}(\tilde{\mathbf{y}}, \mathbf{y}) = \frac{1}{K} \sum_{i=1}^{K} [\![\tilde{\mathbf{y}}[i] \neq \mathbf{y}[i]]\!]$.

[12] show that the two loss functions focus on different statistics of the underlying probability distribution from a Bayesian perspective. While a wide range of other loss functions exist [13], in this paper we only focus on $0/1$ and Hamming because they connect tightly with the ECC framework that will be discussed.[1]

B. Related Works

The hypercube view [11] unifies many existing problem transformation approaches [13], which transform multi-label classification into one or more reduced learning tasks. For instance, one simple problem transformation approach is called Binary Relevance (BR), which learns one binary classifier per individual label. Another simple problem transformation approach is called label powerset (LP), which transforms multi-label classification to one multi-class classification task with a huge number of extended labels. One popular problem transformation approach that lies between BR and LP is called random k-label sets (RAKEL) [13], which transforms multi-label classification into many multi-class classification tasks with a smaller number of extended labels.

Some existing problem transformation approaches focus on *compressing* the label-set vector $\mathbf{y}$ [11], [14]—removing the redundancy within the binary signals (label sets) to form shorter codewords—which follows a classic task in information theory based on Shannon's first theorem [3]. Another classic task in information theory aims at *expansion*—adding redundancy in the (longer) codewords to ensure robust decoding against noise contamination. The power of expansion is characterized by Shannon's second theorem [3]. The error-correcting code (ECC) targets towards using the power of expansion systematically. In particular, the ECC works by encoding a block of signal to a longer codeword $\mathbf{b}$ before passing it through the noisy channel and then decoding the received codeword $\tilde{\mathbf{b}}$ back to the block appropriately. Then, under some assumptions [15], the block can be perfectly recovered—resulting in zero block-decoding error; in some cases, the block can only be almost perfectly recovered—resulting in a few bit-decoding errors.

If we take the "block" as the label set $\mathbf{y}$ for every example $(\mathbf{x}, \mathbf{y})$ and a batch of base learners as a channel that outputs the contaminated block $\tilde{\mathbf{b}}$, the block-decoding error corresponds to $\Delta_{0/1}$ while the bit-decoding error corresponds to a scaled version of Δ_{HL}. Such a correspondence motivates us to study whether suitable ECC designs can be used to improve multi-label classification, which will be formalized in Section II.

Most of the commonly-used ECC in communication systems are binary ECC. That is, the codeword $\mathbf{b}$ is a binary vector. We are going to apply this kind of ECC on multi-label classification, and will review some of them in Section II-A. Another kind of ECC is real-valued ECC, which uses real-valued vectors as the codewords. [16] and [17] take this direction and design special encoding and decoding functions for multi-label classification. [16] uses canonical correlation analysis to find the most linearly-predictable transform of the original labels; [17] uses metric learning to locate a good encoding function. Both works take approximate Bayesian inference for decoding.

II. ML-ECC Framework

We now describe the proposed ECC framework in detail. The main idea is to use an ECC encoder $enc(\cdot)\colon \{0,1\}^K \to \{0,1\}^M$ to expand the original label set $\mathbf{y} \in \{0,1\}^K$ to a codeword $\mathbf{b} \in \{0,1\}^M$ that contains redundant information. Then, instead of learning a multi-label classifier $h(\mathbf{x})$ between $\mathbf{x}$ and $\mathbf{y}$, we learn a multi-label classifier $\tilde{h}(\mathbf{x})$ between $\mathbf{x}$ and the corresponding $\mathbf{b}$. In other words, we transform the original multi-label classification problem into another (larger) multi-label classification task. During prediction, we use $h(\mathbf{x}) = dec \circ \tilde{h}(\mathbf{x})$, where $dec(\cdot)\colon \{0,1\}^M \to \{0,1\}^K$ is the corresponding ECC decoder, to get a multi-label prediction $\tilde{\mathbf{y}} \in \{0,1\}^K$. The simple steps of the framework are shown as follows:

- Parameter: an ECC with encoder $enc(\cdot)$ and decoder $dec(\cdot)$; a base multi-label learner $\mathcal{A}_b$
- Training: Given $D = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$,
  1) ECC-encode each $\mathbf{y}_n$ to $\mathbf{b}_n = enc(\mathbf{y}_n)$;
  2) Return $\tilde{h} = \mathcal{A}_b\Big(\big\{\big(\mathbf{x}_n, \mathbf{b}_n\big)\big\}\Big)$.
- Prediction: Given any $\mathbf{x}$,
  1) Predict a codeword $\tilde{\mathbf{b}} = \tilde{h}(\mathbf{x})$;
  2) Return $h(\mathbf{x}) = dec(\tilde{\mathbf{b}})$ by ECC-decoding.

This algorithm is simple and general. It can be coupled with any block-coding ECC and any base learner $\mathcal{A}_b$ to form a new multi-label classification algorithm. For instance, the ML-BCHRF method [7] uses the BCH code (see Subsection II-A3) as the ECC and BR on Random Forest as the base learner $\mathcal{A}_b$. Note that [7] did not describe why ML-BCHRF may lead to improvements in multi-label classification. Next, we show a simple theorem that connects the ECC framework with $\Delta_{0/1}$.

[1] We follow the final remark of [12] to only focus on the loss functions that are related to our algorithmic goals.

Many ECCs can guarantee to correct up to m bit flipping errors in a codeword of length M. We will introduce some of those ECC in Section II-A. Then, if Δ_{HL} of $\tilde{h}$ is low, the ECC framework guarantees that $\Delta_{0/1}$ of h is low. The guarantee is formalized as follows.

Theorem 1: Consider an ECC that can correct up to m bit errors in a codeword of length M. Then, for any T test examples $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$, let $\mathbf{b}_t = enc(\mathbf{y}_t)$. If

$$\Delta_{HL}(\tilde{h}) = \frac{1}{T}\sum_{t=1}^{T} \Delta_{HL}(\tilde{h}(\mathbf{x}_t), \mathbf{b}_t) \leq \epsilon,$$

then $h = dec \circ \tilde{h}$ satisfies

$$\Delta_{0/1}(h) = \frac{1}{T}\sum_{t=1}^{T} \Delta_{0/1}(h(\mathbf{x}_t), \mathbf{y}_t) \leq \frac{M\epsilon}{m+1} .$$

Proof: When the average Hamming loss of $\tilde{h}$ is at most ϵ, $\tilde{h}$ makes at most ϵTM bits of error on all $\mathbf{b}_t$. Since the ECC corrects up to m bits of errors in one $\mathbf{b}_t$, an adversarial has to make at least $m+1$ bits of errors on $\mathbf{b}_t$ to make $h(\mathbf{x}_t)$ different from $\mathbf{y}_t$. The number of such $\mathbf{b}_t$ can be at most $\frac{\epsilon TM}{m+1}$. Thus, $\Delta_{0/1}(h)$ is at most $\frac{\epsilon TM}{T(m+1)}$. ■

From Theorem 1, it appears that we should simply use some stronger ECC, for which m is larger. Nevertheless, note that we are applying the ECC in a learning scenario. Thus, ϵ is not a fixed value, but depends on whether $\mathcal{A}_b$ can learn well from $\tilde{D}$. Stronger ECC usually contains redundant bits that come from complicated compositions of the original bits in $\mathbf{y}$, and the compositions may not be easy to learn. The trade-off has been revealed when applying the ECC to multi-class classification [6]. Next, we study the ECC with different strength and empirically verify the trade-off in Section IV.

A. Review of Classic ECC

Next, we review four ECC designs that will be used in the empirical study. The four designs cover a broad spectrum of practical choices in terms of strength.

1) Repetition Code: One of the simplest ECCs is repetition code (REP) [15], for which every bit in $\mathbf{y}$ is repeated $\lfloor \frac{M}{K} \rfloor$ or $\lfloor \frac{M}{K} \rfloor + 1$ times in $\mathbf{b}$ during encoding. The decoding takes a majority vote using the received copies of each bit. Because of the majority vote, repetition code corrects up to $m_{REP} = \lfloor \frac{M}{2K} - \frac{1}{2} \rfloor$ bit errors in $\mathbf{b}$. We will discuss the connection between REP and the RAKEL algorithm in Section II-B.

2) Hamming on Repetition Code: A slightly more complicated ECC than REP is called the Hamming code (HAM) [18], which can correct $m_{HAM} = 1$ bit error in $\mathbf{b}$ by adding some parity check bits (exclusive-OR operations of some bits in $\mathbf{y}$). One typical choice of HAM is HAM$(7, 4)$, which encodes any $\mathbf{y}$ with $K = 4$ to $\mathbf{b}$ with $M = 7$. Note that $m_{HAM} = 1$ is worse than m_{REP} when M is large. Thus, we consider applying HAM$(7, 4)$ on every 4 (permuted) bits of REP. That is, to form a codeword $\mathbf{b}$ of M bits from a block $\mathbf{y}$ of K bits, we first construct an REP of $4\lfloor M/7 \rfloor + (M \bmod 7)$ bits from $\mathbf{y}$; then for every 4 bits in the REP, we add 3 parity bits to $\mathbf{b}$ using HAM$(7, 4)$. The resulting code will be named Hamming on Repetition (HAMR). During decoding, the decoder of HAM$(7, 4)$ is first used to recover the 4-bit sub-blocks in the REP. Then, the decoder of REP (majority vote) takes place.

It is not hard to compute m_{HAMR} by analyzing the REP and HAM parts separately. When M is a multiple of 7, $m_{HAMR} = 2 \cdot \lfloor \frac{2M}{7K} - \frac{1}{2} \rfloor$, which is generally better than m_{REP} especially when $\frac{M}{K}$ is large. Thus, HAMR is slightly stronger than REP for ECC purposes. We include HAMR in our study to verify whether a simple inclusion of some parity bits for the ECC can readily improve the performance for multi-label classification.

3) Bose-Chaudhuri-Hocquenghem Code: BCH [19], [20] code can be viewed as a sophisticated extension of HAM and allows correcting multiple bit errors. BCH with length $M = 2^p - 1$ has $(M - K)$ parity bits, and it can correct $m_{BCH} = \frac{M-K}{p}$ bits of error [15], which is in general stronger than REP and HAMR. The caveat is that the decoder of BCH is more complicated than the ones of REP and HAMR.

We include BCH in our study because it is one of the most popular ECCs in real-world communication systems. In addition, we compare BCH with HAMR to see if a strong ECC can do better for multi-label classification.

4) Low-density Parity-check Code: Low-density parity-check code (LDPC) [15] is recently drawing much research attention in communications. LDPC shares an interesting connection between ECC and Bayesian learning [15]. While it is difficult to state the strength of LDPC in terms of a single m_{LDPC}, LDPC has been shown to approach the theoretical limit in some special channels [21], which makes it a state-of-the-art ECC. We choose to include LDPC in our study to see whether it is worthwhile to go beyond BCH with more sophisticated encoder/decoders.

B. ECC View of RAKEL

RAKEL is a multi-label classification algorithm proposed by [13]. Define a k-label set as a size-k subset of $\mathcal{L}$. Each iteration of RAKEL randomly selects a (different) k-label set and builds a multi-label classifier on the k labels with a Label Powerset (LP). After running for R iterations, RAKEL obtains a size-R ensemble of LP classifiers. The prediction on each label is done by a majority vote from classifiers associated with the label.

Equivalently, we can draw (with replacement) $M = Rk$ labels first before building the LP classifiers. Then, selecting k-label sets is equivalent to encoding $\mathbf{y}$ by a variant of REP, which will be called RAKEL repetition code (RREP). Similar to REP, each bit $\mathbf{y}[i]$ is repeated several times in $\mathbf{b}$ since label i is involved in several k-label sets. After encoding $\mathbf{y}$ to $\mathbf{b}$, each LP classifier, called k-powerset, acts as a sub-channel that transmits a size-k sub-block of the codeword $\mathbf{b}$. The prediction procedure follows the decoder of the usual REP.

The ECC view above decomposes the original RAKEL into two parts: the ECC and the base learner $\mathcal{A}_b$. We will empirically study how the two parts affect the performance of multi-label classification in Section IV.

C. Comparison to Real-valued Codewords

As mentioned in Section I-B, [16] and [17] propose using real-valued ECC for multi-label classification. There are two

major differences between the real-valued ECC and the binary ECC. In terms of codewords, the real-valued codewords of [16] and [17] are linear combinations of labels. On the other hand, the binary codewords introduced in Section II-A are parity checks (exclusive-OR) of labels, which is a linear combination of labels under Galois field GF_2.

In terms of base learners, the transformed learning tasks of real-valued codewords are regression instead of classification. In [16] and [17], the regression task of each codeword bit is learned separately, which is like applying the BR base learner. On the other hand, the transformed learning task of binary codewords can be learned with other base learners such as k-powerset.

III. Geometric Decoder for Linear ECC

The real-valued codewords motivate us to design a new decoder for binary ECC in the ML-ECC framework. The goal of the new decoder is to utilize the channel measurement information, i.e. the real-valued confidence for the bit to be 1. Such information is available to the decoder when using the BR base learner (channel) with probabilistic outputs. The real-valued confidence may help improve the performance of the ML-ECC framework by focusing more on the highly-confident bits.

The off-the-shelf ECC decoders usually do not exploit the channel measurement information, but take advantages of the algebraic structure of the ECC to locate possible bit errors. From the hypercube view, they decode a vertex of $\{0,1\}^M$ (binary prediction on codewords) to a vertex of $\{0,1\}^K$ (binary prediction on labels). The proposed decoder, on the contrary, utilizes the information and takes the geometry of the hypercube into account to perform interior-to-interior decoding from $[0,1]^M$ to $[0,1]^K$. That is, the proposed decoder is a soft-input soft-output decoder. The soft input bits contain the channel measurement information, and the value of each bit represents the confidence in the bit being 1. The soft prediction of labels are the confidence in whether the label presents. For evaluation, the soft predictions are then rounded to $\{0,1\}^K$ as in [11]. We call the proposed decoder *geometric decoder*, and call the off-the-shelf decoders *algebraic decoders*.

Here, we focus on linear codes, whose encoding function can be written as a matrix-vector multiplication under Galois field GF_2. All the repetition code, Hamming code, BCH code, and LDPC code are linear codes. Let G be the generating matrix of a linear code, $g_{ij} \in \{0,1\}$. The encoding is done by $\mathbf{b} = enc(\mathbf{y}) = G \cdot \mathbf{y} \pmod 2$, or equivalently we may write the formula in terms of exclusive-OR (XOR) operations:

$$b_i = \bigoplus_{j:g_{ij}=1} y_j$$

That is, the codeword bit b_i is the result of XOR of some label bits y_j. The XOR operations are equivalent to multiplications if we map $1 \to -1$ and $0 \to 1$. By defining $\hat{b}_i = 1 - 2b_i$ and $\hat{y}_j = 1 - 2y_j$, the encoding can also be written as

$$\hat{b}_i = \prod_{j:g_{ij}=1} \hat{y}_j$$

We denote this form as *multiplication encoding*.

It is difficult to generalize the XOR operation from binary to real values, but multiplication by itself can be defined on real values. We take this advantage and use it to form our geometric decoder. Our geometric decoder would find the $\tilde{\mathbf{y}}$ that minimizes the L_2 distance between $\tilde{\mathbf{b}}$ and the multiplication encoding result of the $\tilde{\mathbf{y}}$:

$$dec_{geo}(\tilde{\mathbf{b}}) = \operatorname*{argmin}_{\tilde{\mathbf{y}} \in [0,1]^K} \sum_{i=1}^{M} \left((1 - 2\tilde{b}_i) - \prod_{j:g_{ij}=1} (1 - 2\tilde{y}_j) \right)^2$$

Note that the squared L_2 distance between codewords is an approximation of the Hamming distance in binary space $\{0,1\}^M$.

For repetition code, since only one y_j is considered for each b_i, the optimal solution of the problem would be the same as averaging over the predictions on the same label for each label. However, for general linear codes, it is difficult to find the global optimum since the optimization problem may not be convex. Instead, we may apply a variant of coordinate descent optimization to find a local minimum. That is, in each step we optimize only one $\tilde{y}_j$ while fixing other $\tilde{y}_j$. To optimize one $\tilde{y}_j$, we only have to solve a second-order single-variable optimization problem, which enjoys an efficient analytic solution.

The benefit of using soft output geometric decoder is that the multiplication-approximated XOR preserves some geometric information. That is, close points in $[0,1]^K$ would also be close after multiplication encoding. Moreover, approximating XOR by multiplication allows us to consider soft input, i.e. confidence information, during decoding.

IV. Experiments

First we compare RREP, HAMR, BCH, and LDPC with the ML-ECC framework on seven real-world datasets in different domains: `scene`, `emotions`, `yeast`, `tmc2007`, `genbase`, `medical`, and `enron` [22] using the algebraic decoder. The number of classes (K) is shown in Table I. All the results are reported with the mean and standard error on random splitting test set over 30 runs. The sizes of training and testing sets are set according to the sizes in original datasets. Note that for `tmc2007` dataset, which contains 28596 instances in total, we randomly sample 5% for training and another 5% for testing in each run.

We set RREP with $k = 3$. Then, for each ECC, we first consider a 3-powerset with either Random Forest, non-linear support vector machine (SVM), or logistic regression as the multi-class classifier inside the 3-powerset. Note that we randomly permute the bits of $\mathbf{b}$ and apply an inverse permutation on $\tilde{\mathbf{b}}$ for those ECC other than RREP to ensure that each 3-powerset works on diverse sub-blocks. In addition to the 3-powerset base learners, we also consider BR base learners in Subsection IV-D.

We take the default Random Forest from Weka [23] with 60 trees. For the non-linear SVM, we use LIBSVM [24] with the Gaussian kernel and choose (C, γ) by cross validation on training data from $\{2^{-5}, 2^{-3}, \cdots, 2^{7}\} \times \{2^{-9}, 2^{-7}, \cdots, 2^{1}\}$.

In addition, we use LIBLINEAR [25] for the logistic regression and choose the parameter C by cross validation from $\{2^{-5}, 2^{-3}, \cdots, 2^{7}\}$.

A. Validity of ML-ECC Framework

First, we demonstrate the validity of the ML-ECC framework. We fix the codeword length M to about 20 times larger than the number of labels K. The numbers are in the form $2^p - 1$ for integer p because the BCH code only works on such lengths. More experiments on different codeword lengths are presented in Section IV-B. Here the base multi-label learner is the 3-powerset with Random Forests. Following the description in Section II-B, RREP with the 3-powerset is exactly the same as RAKEL with $k = 3$.

The results on $0/1$ loss is shown in Figure 1(a). HAMR achieves lower $\Delta_{0/1}$ than RREP on 5 out of the 7 datasets (`scene`, `emotions`, `yeast`, `tmc2007`, and `medical`) and achieves similar $\Delta_{0/1}$ with RREP on the other 2. This verifies that using some parity bits instead of repetition improves the strength of ECC, which in turn improves the $0/1$ loss. Along the same direction, BCH performs even better than both HAMR and RREP, especially on `medical` dataset. The superior performance of BCH justifies that the ECC is useful for multi-label classification. On the other hand, another sophisticated code, LDPC, gets higher $0/1$ loss than BCH on every dataset, and even higher $0/1$ loss than RREP on the `emotions` and `yeast` datasets, which suggest that LDPC may not be a good choice for the ECC framework.

Next we look at Δ_{HL} shown in Figure 1(b). The Hamming loss of HAMR is comparable to that of RREP, where each wins on two datasets. BCH beats both HAMR and RREP on the `tmc2007`, `genbase`, and `medical` datasets but loses on the other four datasets. LDPC has the highest Hamming loss among the codes on all datasets. Thus, simpler codes like RREP and HAMR perform better in terms of Δ_{HL}. A stronger code like BCH may guard $\Delta_{0/1}$ better, but it can pay more in terms of Δ_{HL}.

Similar results show up when using the Gaussian SVM or logistic regression as the base learner instead of Random Forest, as shown in Tables I and II. The boldface entries are the lowest-loss ones for the given dataset and base learner. The results validate that the performance of multi-label classification can be improved by applying the ECC. More specifically, we may improve the RAKEL algorithm by learning some parity bits instead of repetitions. Based on this experiment, we suggest that using HAMR for multi-label classification will improve the $\Delta_{0/1}$ while maintaining comparable Δ_{HL} with RAKEL. If we use BCH instead, we will improve $\Delta_{0/1}$ further but may pay for Δ_{HL}.

B. Comparison of Codeword Length

Now, we compare on the length of codewords M. With larger M, the codes can correct more errors but the base learners have to take longer time to train. By experimenting different M, we may find a better trade-off between performance and efficiency.

The performance of the ECC framework with different codeword lengths on the `scene` dataset is shown on Figure 2. Here, the base learner is again the 3-powerset with Random Forests. The codeword length M varies from 31 to 127, which is about 5 to 20 times of number of labels L. We do not include shorter codewords because their performance are not stable. Note that BCH only allows $M = 2^p - 1$ and thus we conduct experiments of BCH on those codeword lengths.

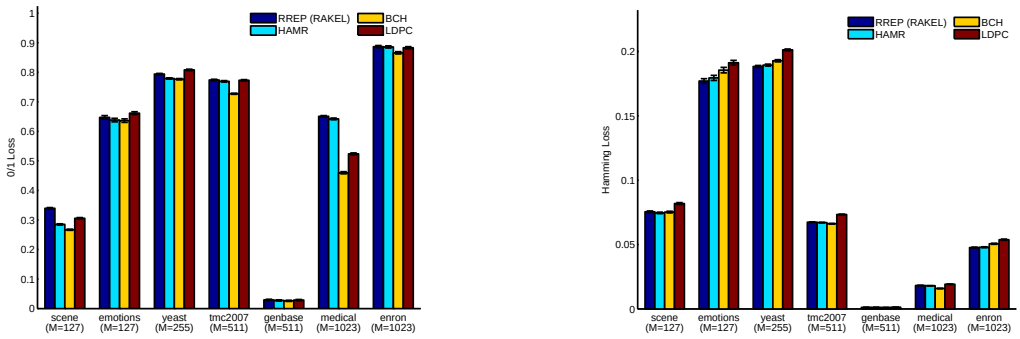
We first look at the $0/1$ loss in Figure 2(a). The horizontal axis indicates the codeword length M and the vertical axis is the $0/1$ loss on the test set. We see that $\Delta_{0/1}$ of RREP stays around 0.335 no matter how long the codewords are. This implies that the power of repetition bits reaches its limit very soon. For example, when all the 3-powerset combinations of labels are learned, additional repetitions give very limited improvements. Therefore, methods using repetition bits only, such as RAKEL, cannot take advantage from the extra bits in the codewords.

The $\Delta_{0/1}$ of HAMR and BCH are slightly decreasing with M, but the differences between $M = 63$ and $M = 127$ are generally small (smaller than the differences between $M = 31$ and $M = 63$, in particular). This indicates that learning some parity bits provides additional information for prediction, which cannot be learned easily from repetition bits, and such information remains beneficial for longer codewords, comparing to repetition bits. One reason is that the number of 3-powerset combinations of parity bits is exponentially more than that of combinations of labels. The performance of LDPC is not as stable as the other codes, possibly because of its sophisticated decoding step. Somehow, we still see that its $\Delta_{0/1}$ decreases slightly with M.

Figure 2(b) shows Δ_{HL} versus M for each ECC. The Δ_{HL} of RREP is the lowest among the codes when M is small, but it remains almost constant when $M \geq 63$, while Δ_{HL} of HAMR and BCH are still decreasing. This matches our finding that extra repetition bits give limited information. When $M = 127$, BCH is comparable to RREP in terms of Δ_{HL}. HAMR is even better than RREP at that codeword length, and becomes the best code regarding Δ_{HL}. Thus, while a stronger code like BCH may guard $\Delta_{0/1}$ better, it can pay more in terms of Δ_{HL}.

As stated in Sections I-A and II, the base learners serve as the channels in the ECC framework and the performance of base learners may be affected by the codes. Therefore, using a strong ECC does not always improve multi-label classification performance. Next, we verify the trade-off by measuring the bit error rate Δ_{BER} of $\tilde{h}$, which is defined as the Hamming loss between the predicted codeword $\tilde{h}(\mathbf{x})$ and the actual codeword $\mathbf{b}$. Higher bit error rate implies that the transformed task is harder.

Figure 2(c) shows the Δ_{BER} versus M for each ECC. RREP has almost constant bit error rate. HAMR also has nearly constant bit error rate but at a higher value. The bit error rate of BCH is similar to that of HAMR when the codeword is short, but the bit error rate increases with M. One explanation is that some of the parity bits are harder to learn than repetition bits. The ratio between repetition bits and parity bits of both RREP and HAMR codes is a constant of



(a) (b)

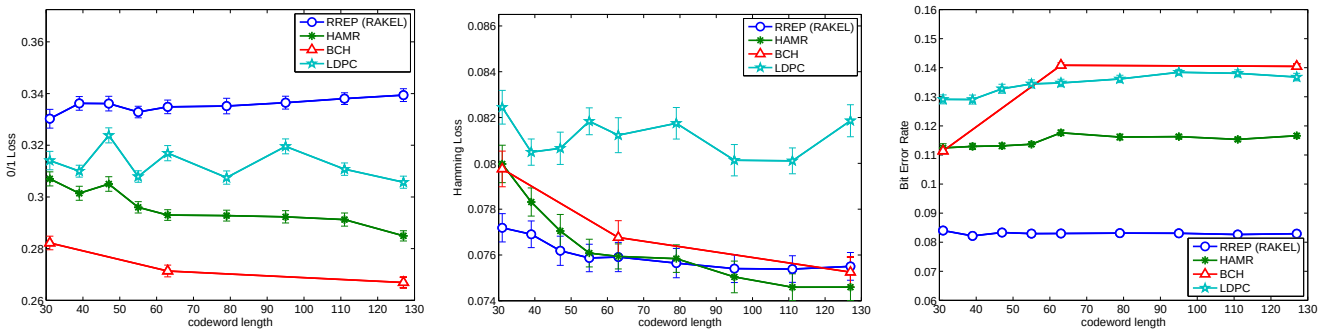
Fig. 1. Performance of ML-ECC using the 3-powerset with Random Forests: (a) $0/1$ loss (b) Hamming loss

TABLE I
$0/1$ LOSS OF ML-ECC USING 3-POWERSET BASE LEARNERS

base learner	ECC	scene (K=6) (M=127)	emotions (K=6) (M=127)	yeast (K=14) (M=255)	tmc2007 (K=22) (M=511)	genbase (K=27) (M=511)	medical (K=45) (M=1023)	enron (K=53) (M=1023)
Random Forests	RREP (RAKEL)	.3394 ± .0025	.6472 ± .0060	.7939 ± .0022	.7738 ± .0025	.0295 ± .0021	.6508 ± .0024	.8866 ± .0038
	HAMR	.2849 ± .0020	**.6381 ± .0060**	.7789 ± .0021	.7693 ± .0024	**.0276 ± .0021**	.6420 ± .0029	.8855 ± .0036
	BCH	**.2669 ± .0020**	**.6361 ± .0059**	**.7764 ± .0021**	**.7273 ± .0018**	**.0263 ± .0020**	**.4598 ± .0036**	**.8659 ± .0039**
	LDPC	.3057 ± .0023	.6616 ± .0048	.8080 ± .0024	.7728 ± .0022	.0288 ± .0021	.5238 ± .0032	.8830 ± .0036
Gaussian SVM	RREP (RAKEL)	.2856 ± .0016	**.7759 ± .0055**	.7601 ± .0023	.7196 ± .0024	.0295 ± .0025	.3679 ± .0036	.8725 ± .0041
	HAMR	.2639 ± .0017	**.7736 ± .0050**	.7530 ± .0021	.7162 ± .0023	.0303 ± .0026	.3641 ± .0031	.8693 ± .0042
	BCH	**.2576 ± .0017**	**.7744 ± .0053**	**.7429 ± .0017**	**.7095 ± .0020**	**.0255 ± .0019**	**.3394 ± .0027**	**.8477 ± .0045**
	LDPC	.2780 ± .0020	.8040 ± .0044	.7574 ± .0021	.7403 ± .0019	.0285 ± .0021	.3856 ± .0031	.8666 ± .0041
Logistic Regression	RREP (RAKEL)	.3601 ± .0019	**.6949 ± .0070**	.8161 ± .0017	.7408 ± .0024	.3593 ± .0078	.5507 ± .0254	.8762 ± .0035
	HAMR	.3293 ± .0017	**.6955 ± .0058**	.8061 ± .0019	.7383 ± .0025	.2275 ± .0099	.5268 ± .0230	.8754 ± .0035
	BCH	**.3148 ± .0018**	.7068 ± .0046	**.7899 ± .0020**	**.7233 ± .0024**	**.0250 ± .0018**	**.3797 ± .0044**	**.8504 ± .0042**
	LDPC	.3655 ± .0028	.7295 ± .0056	.8082 ± .0024	.7562 ± .0027	.0325 ± .0018	.4516 ± .0083	.8653 ± .0038

TABLE II
HAMMING LOSS OF ML-ECC USING 3-POWERSET BASE LEARNERS

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random RandoForest	RREP (RAKEL)	.0755 ± .0006	**.1770 ± .0018**	**.1884 ± .0007**	.0674 ± .0003	.0012 ± .0001	.0182 ± .0001	**.0477 ± .0004**
	HAMR	**.0746 ± .0006**	.1795 ± .0020	.1894 ± .0008	.0671 ± .0003	.0012 ± .0001	.0180 ± .0001	**.0479 ± .0004**
	BCH	.0753 ± .0007	.1855 ± .0021	.1928 ± .0008	**.0662 ± .0003**	**.0011 ± .0001**	**.0159 ± .0001**	.0506 ± .0004
	LDPC	.0819 ± .0007	.1912 ± .0019	.2012 ± .0007	.0734 ± .0003	.0013 ± .0001	.0192 ± .0002	.0538 ± .0005
Gaussian SVM	RREP (RAKEL)	**.0719 ± .0005**	**.2432 ± .0021**	**.1853 ± .0007**	**.0613 ± .0003**	.0013 ± .0001	**.0112 ± .0001**	**.0449 ± .0004**
	HAMR	.0724 ± .0005	.2490 ± .0023	.1868 ± .0006	**.0610 ± .0003**	.0013 ± .0001	**.0111 ± .0001**	**.0449 ± .0004**
	BCH	.0739 ± .0006	.2644 ± .0019	.1898 ± .0008	.0629 ± .0003	**.0010 ± .0001**	.0114 ± .0001	.0516 ± .0006
	LDPC	.0755 ± .0006	.2634 ± .0027	.1917 ± .0007	.0679 ± .0003	.0014 ± .0001	.0140 ± .0001	.0530 ± .0005
Logistic Regression	RREP (RAKEL)	**.0915 ± .0005**	**.2026 ± .0025**	**.1993 ± .0007**	**.0634 ± .0003**	.0179 ± .0006	.0190 ± .0011	**.0453 ± .0003**
	HAMR	**.0910 ± .0005**	.2070 ± .0024	.2003 ± .0007	**.0634 ± .0003**	.0102 ± .0005	.0176 ± .0009	**.0454 ± .0003**
	BCH	.0920 ± .0005	.2233 ± .0022	.2051 ± .0008	.0653 ± .0003	**.0013 ± .0001**	**.0137 ± .0003**	.0505 ± .0004
	LDPC	.0989 ± .0007	.2202 ± .0021	.2054 ± .0007	.0701 ± .0003	.0024 ± .0002	.0187 ± .0006	.0528 ± .0004



(a) (b) (c)

Fig. 2. Performance on `scene` vs. codeword length (ML-ECC using the 3-powerset with Random Forests): (a) $0/1$ loss (b) Hamming loss (c) bit-error rate

M (RREP has no parity bits, and HAMR has 3 parity bits for every 4 repetition bits), while BCH has more parity bits with larger M. The different bit error rates justify the trade-off between the strength of the ECC and the hardness of the base learning tasks. With more parity bits, one can correct more bit errors, but may have harder tasks to learn; when using fewer parity bits or even no parity bits, one cannot correct many errors, but will enjoy simpler learning tasks.

Similar results show up in other datasets with all three base learners. The performance on the `yeast` dataset with the 3-powerset and Random Forests is shown in Figure 3. Because the number of labels in the `yeast` dataset is about twice of that in the `scene` dataset, the codeword length here ranges from 63 to 255, which is also about twice longer than that in the experiments on the `scene` dataset. Again, we see that the benefits of parity bits remain valid for longer codewords than repetition bits and that more parity bits cause the transformed task harder to learn. This result points out the trade-off between the strength of the ECC and the hardness of the base learning tasks.

C. Bit Error Analysis

To further analyze the difference between different ECC designs, we zoom in to $M = 127$ of Figure 2. The instances are divided into groups according to the number of bit errors at that instance. The relative frequency of each group, i.e., the ratio of the group size to the total number of instances, is plotted in Figure 4(a). The average $\Delta_{0/1}$ and Δ_{HL} of each group are also plotted in Figure 4(b) and 4(c). The curve of each ECC forms two peak regions in Figure 4(a). Besides the peak at 0, which means no bit error happens on the instances, the other peak varies from one code to another. The positions of the peaks suggest the hardness of the transformed learning task, similar to our findings in Figure 2(c).

We can clearly see the difference on the strength of different ECC from Figure 4(b). BCH can tolerate up to 31-bit errors, but its $\Delta_{0/1}$ sharply increases over 0.8 for 32-bit errors. HAMR can correct 13-bit errors perfectly, and its $\Delta_{0/1}$ increases slowly when more errors occur. Both RREP and LDPC can perfectly correct only 9-bit errors, but LDPC is able to sustain a low $\Delta_{0/1}$ even when there are 32-bit errors. It would be interesting to study the reason behind this long tail from a Bayesian network perspective.

We can also look at the relation between the number of bit errors and Δ_{HL}, as shown in Figure 4(c). The BCH curve grows sharply when the number of bit errors is larger than 31, which links to the inferior performance of BCH over RREP in terms of Δ_{HL}. The LDPC curve grows much slower, but its right-sided peak in Figure 4(a) still leads to higher overall Δ_{HL}. On the other hand, RREP and HAMR enjoy a better balance between the peak position in Figure 4(a) and the growth in Figure 4(c) and thus lower overall Δ_{HL}.

Figure 4(a) suggests that the transformed learning task of more sophisticated ECC is harder. The reason is that sophisticated ECC contains many parity bits, which are the exclusive-or of labels, and the parity bits are harder to learn by the base learners. We demonstrate this in Figure 5 using `scene` dataset (6 labels) and fixing $M = 127$. The codeword bits are divided into groups according to the number of labels XOR'ed to form the bit. The relative frequency of each group is plotted in Figure 5(a). We can see that all codeword bits of RREP are formed by 1 label, and the bits of HAMR are formed by 1 or 3 labels. For BCH and LDPC, the number of labels XOR'ed in the bits may be none (0) to all (6) labels, while most of the bits are the XOR of half of the labels (3 labels).

Next we show how well the base learners learned on each group in Figure 5(b). Here the base learner is 3-powerset with Random Forests. The figure suggests that the parity bits (XOR'ing 2 or more labels) result in harder learning tasks and higher bit error rates than original labels (XOR'ing 1 label). One exception is the bits XOR'ed from all (6) labels, which is easier to learn than original labels. The reason is that the bit XOR'ed from all labels is equivalent to the indicator of odd number of labels, and a constant predictor works well for this because in the `scene` dataset about 92% of all instances has 1 or 3 labels. Since BCH and LDPC have many bits XOR'ed from 2-4 labels, their bit error rates are higher than RREP and HAMR as shown in Figure 2(c).

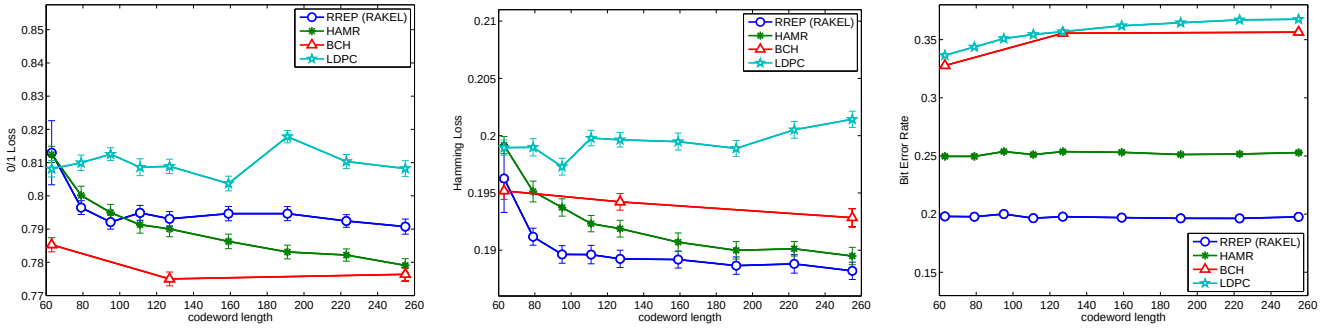
These findings also appear on other datasets and other base learners, such as `medical` dataset (45 labels, $M = 1023$) shown in Figure 6. BCH and LDPC have many bits XOR'ed from about half of the labels, and the transformed learning tasks of such bits are harder to learn than that of original labels.

D. Comparison with Binary Relevance

In addition to the 3-powerset base learners, we also consider BR base learners, which simply build a classifier for each bit in the codeword space. Note that if we couple the ECC framework with RREP and BR, the resulting algorithm is almost the same as the original BR. For example, using RREP and BR with SVM is equivalent to using BR with bootstrap aggregated SVM.

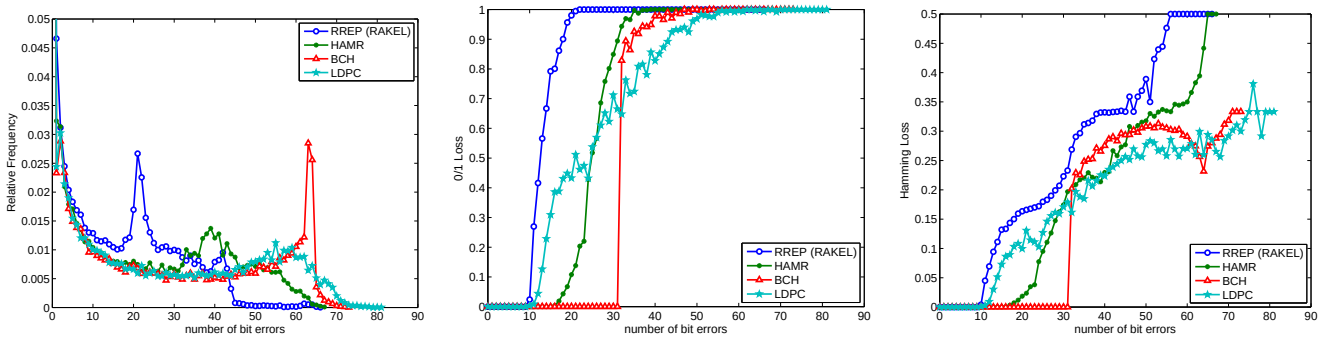
We first compare the performance between the ECC designs using the BR base learner with Random Forests. The result on $0/1$ loss is shown in Figure 7(a). From the figure, we can see that BCH and HAMR reaches superior performance to other ECC, with BCH being a better choice. RREP (BR), on the other hand, leads to the worst $0/1$ loss. The result again justifies the usefulness of coupling BR with the ECC instead of only the original $\mathbf{y}$. Note that LDPC also performs better than BR on two datasets, but is not as good as HAMR and BCH. Thus, over-sophisticated ECC like LDPC may not be necessary for multi-label classification.

In Figure 7(b), we present the results on Δ_{HL}. In contrast to the case when using the 3-powerset base learner, here both HAMR and BCH can achieve better Δ_{HL} than RREP (BR) in most of the datasets. HAMR wins on three datasets, while BCH wins on four. Thus, coupling stronger ECC with the BR base learner can improve both $\Delta_{0/1}$ and Δ_{HL}. However, LDPC performs worse than BR in term of Δ_{HL}, which again shows that LDPC may not be suitable for multi-label classification.



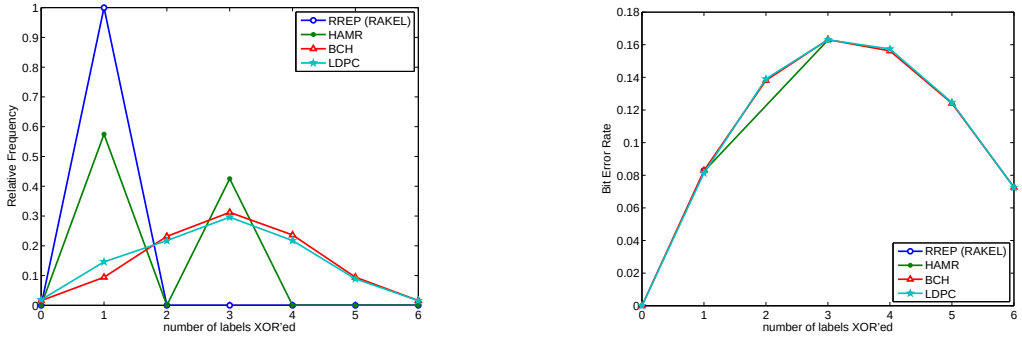
(a) (b) (c)

Fig. 3. Performance on `yeast` vs. codeword length (ML-ECC using the 3-powerset with Random Forests): (a) $0/1$ loss (b) Hamming loss (c) bit-error rate



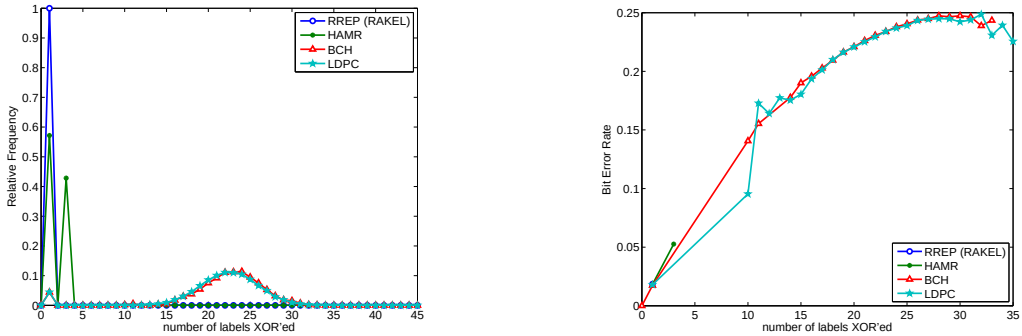
(a) (b) (c)

Fig. 4. Bit errors and losses on the `scene` dataset with $M = 127$: (a) relative frequency (b) $0/1$ loss (c) Hamming loss vs. number of bit errors



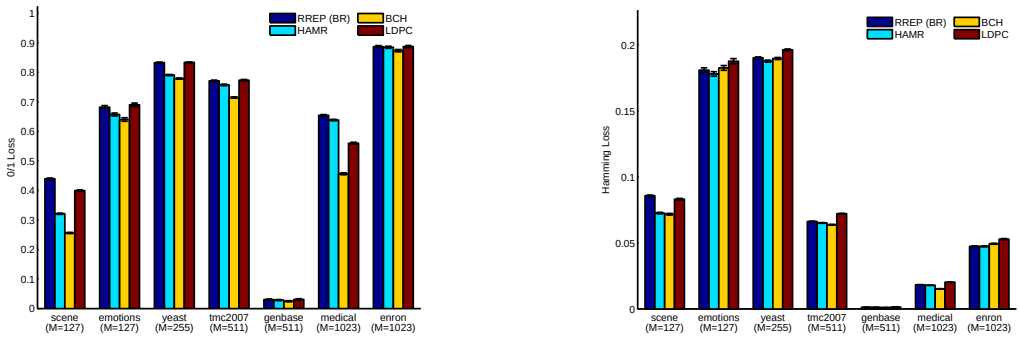
(a) (b)

Fig. 5. Parity bits on the `scene` dataset, 6 labels, 127-bit codeword: (a) relative frequency (b) bit error rate vs. number of labels XOR'ed



(a) (b)

Fig. 6. Parity bits on the `medical` dataset, 45 labels, 1023-bit codeword: (a) relative frequency (b) bit error rate vs. number of labels XOR'ed



(a) (b)

Fig. 7. Performance of ML-ECC using Binary Relevance with Random Forests: (a) $0/1$ loss (b) Hamming loss

Experiments with other base learners also support similar findings, as shown in Tables III and IV. Notice that HAMR performs better than BCH when using Gaussian SVM base learners. Thus, extending BR by learning some more parity bits and decoding them suitably by the ECC is a superior algorithm over the original BR.

Comparing Tables I and III, we see that using 3-powerset achieves lower $0/1$ loss than using BR in most of the cases. However, in terms of Δ_{HL}, as shown in Tables II and IV, there is no clear winner between the 3-powerset and BR.

E. Validity of the Geometric Decoder

The previous experiment shows that using either HAMR or BCH code improves the performance of Binary Relevance learners. Now we are going to examine whether the proposed geometric decoder can further improve the result. We empirically compare the off-the-shelf algebraic decoder with hard inputs, the proposed geometric decoder with hard inputs and the proposed geometric decoder with soft inputs. Note that hard inputs are the direct binary bits in the codeword, and soft inputs contain the confidence that the bits are 1. We include the hard-input geometric decoder into comparison to see whether the geometric decoder is competitive to the algebraic ones when the base learner does not provide soft inputs.

When using BR base learners, Random Forests from WEKA, Gaussian SVM from LIBSVM, and logistic regression from LIBLINEAR all support outputting the confidence for binary classification, which is naturally taken as the soft input.

The results on the BCH code using the Gaussian SVM base learner is shown in Figure 8. We can see from Figure 8(a) that in terms of $\Delta_{0/1}$, both geometric decoders are significantly better than the algebraic one, especially on `emotions` and `yeast` dataset, and the soft-input geometric decoder is slightly better than or similar to the hard-input one.

From Figure 8(b), we can see that the soft-input geometric decoder is much better than the hard-input one in terms of Δ_{HL}, but the algebraic decoder is usually the best here. The reason may be that the geometric decoder minimizes the distance between approximated $enc(\tilde{\mathbf{y}})$ and $\tilde{\mathbf{b}}$ in the codeword space. However, the BCH code does not preserve the Hamming distance during encoding and decoding between $\{0,1\}^K$ and $\{0,1\}^M$, so the geometric decoder, which minimizes the distance in $[0,1]^M$ (and approximately in $\{0,1\}^M$), may not be suitable to the Hamming loss (Hamming distance in $\{0,1\}^K$). But when taking confidence information as soft input, the geometric decoder can perform better on Hamming loss, and become comparable to the algebraic decoder sometimes.

The result shows that the proposed geometric decoder on BCH code is better than the ordinary algebraic decoder for $\Delta_{0/1}$, but not for Δ_{HL}. Also, the soft input is useful for the geometric decoder in terms of both $\Delta_{0/1}$ and Δ_{HL}.

Next we look at the HAMR code. On $0/1$ loss shown in Figure 9(a), the geometric decoders are better than the algebraic one except on the `genbase` and `enron` datasets where all decoders have similar $0/1$ loss. Among the hard-input and soft-input geometric decoders, there is no clear winner. In terms of Hamming loss in Figure 9(b), the soft-input geometric decoder performs the best. Its Hamming loss is significantly lower than the that of the hard-input geometric decoder and the algebraic decoder on `scene`, `emotions`, and `tmc2007` dataset, and similar to them on other datasets. Comparing the hard-input geometric and algebraic decoders, there is no significant difference.

The result again shows that the proposed geometric decoder on HAMR code is better than the ordinary algebraic decoder for $\Delta_{0/1}$, but not for Δ_{HL}. But with soft input, the geometric decoder can give lower Δ_{HL}.

Similar results show up when using other base learners, as shown in Table V and VI. The bold-face entries are the best entries on each dataset given the ECC and base learner. From this experiment we see that using the hard-input geometric decoder instead of the off-the-shelf algebraic one leads to improvement on $\Delta_{0/1}$, but pay for Δ_{HL}. If using the soft-input geometric decoder, the harm of Δ_{HL} is mitigated and the improvement on $\Delta_{0/1}$ is strengthened. Therefore, the soft-input geometric decoder is a better choice for the ML-ECC framework.

F. Comparison with Real-valued ECC

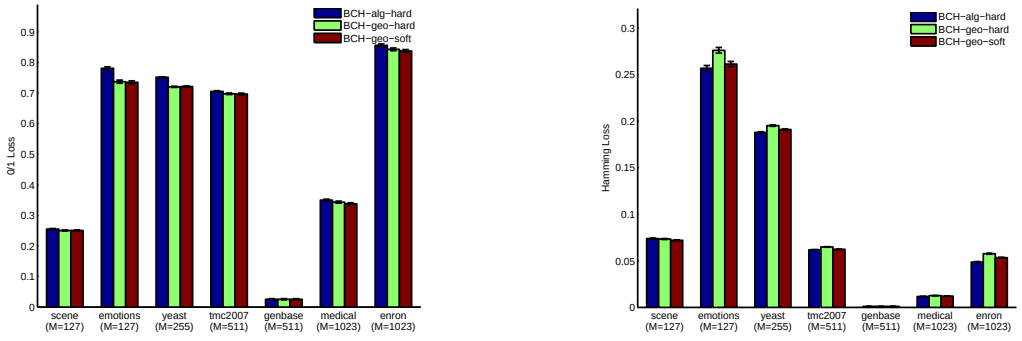
Our ML-ECC framework only considers binary ECC. Here, we compare our ML-ECC framework with real-valued ECC methods: coding with canonical correlation analysis (CCA-OC) [16] and max-margin output coding (MaxMargin) [17], as discussed in Section II-C.

TABLE III
$0/1$ LOSS OF ML-ECC USING BR BASE LEARNERS

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	RREP (BR)	$.4398 \pm .0023$	$.6822 \pm .0055$	$.8332 \pm .0016$	$.7715 \pm .0023$	$.0303 \pm .0020$	$.6546 \pm .0025$	$.8872 \pm .0036$
	HAMR	$.3212 \pm .0021$	$.6574 \pm .0050$	$.7910 \pm .0020$	$.7578 \pm .0025$	$.0288 \pm .0019$	$.6387 \pm .0025$	$.8851 \pm .0036$
	BCH	$\mathbf{.2562 \pm .0020}$	$\mathbf{.6404 \pm .0060}$	$\mathbf{.7792 \pm .0019}$	$\mathbf{.7149 \pm .0022}$	$\mathbf{.0250 \pm .0019}$	$\mathbf{.4567 \pm .0034}$	$\mathbf{.8737 \pm .0038}$
	LDPC	$.3995 \pm .0026$	$.6903 \pm .0058$	$.8338 \pm .0015$	$.7735 \pm .0024$	$.0312 \pm .0022$	$.5601 \pm .0032$	$.8876 \pm .0035$
Gaussian SVM	RREP (BR)	$.3376 \pm .0023$	$.8414 \pm .0052$	$.7955 \pm .0017$	$.7281 \pm .0025$	$.0273 \pm .0021$	$.3721 \pm .0037$	$.8720 \pm .0041$
	HAMR	$.2876 \pm .0018$	$.8073 \pm .0043$	$.7681 \pm .0022$	$.7215 \pm .0025$	$\mathbf{.0243 \pm .0022}$	$.3675 \pm .0036$	$.8718 \pm .0042$
	BCH	$\mathbf{.2552 \pm .0018}$	$\mathbf{.7809 \pm .0050}$	$\mathbf{.7515 \pm .0015}$	$\mathbf{.7053 \pm .0024}$	$\mathbf{.0255 \pm .0019}$	$\mathbf{.3499 \pm .0030}$	$\mathbf{.8561 \pm .0043}$
	LDPC	$.3161 \pm .0022$	$.8523 \pm .0042$	$.7963 \pm .0018$	$.7515 \pm .0023$	$\mathbf{.0243 \pm .0017}$	$.4226 \pm .0034$	$.8782 \pm .0037$
Logistic Regression	RREP (BR)	$.4821 \pm .0024$	$.7396 \pm .0049$	$.8531 \pm .0016$	$.7458 \pm .0026$	$.5084 \pm .0068$	$.5784 \pm .0282$	$.8759 \pm .0035$
	HAMR	$.4050 \pm .0020$	$.7175 \pm .0054$	$.8282 \pm .0015$	$.7405 \pm .0024$	$.3509 \pm .0089$	$.5499 \pm .0247$	$.8740 \pm .0036$
	BCH	$\mathbf{.3291 \pm .0020}$	$\mathbf{.6982 \pm .0048}$	$\mathbf{.8094 \pm .0020}$	$\mathbf{.7205 \pm .0022}$	$\mathbf{.0295 \pm .0018}$	$\mathbf{.4022 \pm .0076}$	$\mathbf{.8579 \pm .0038}$
	LDPC	$.4659 \pm .0028$	$.7507 \pm .0056$	$.8565 \pm .0019$	$.7694 \pm .0027$	$.0528 \pm .0031$	$.5396 \pm .0149$	$.8795 \pm .0036$

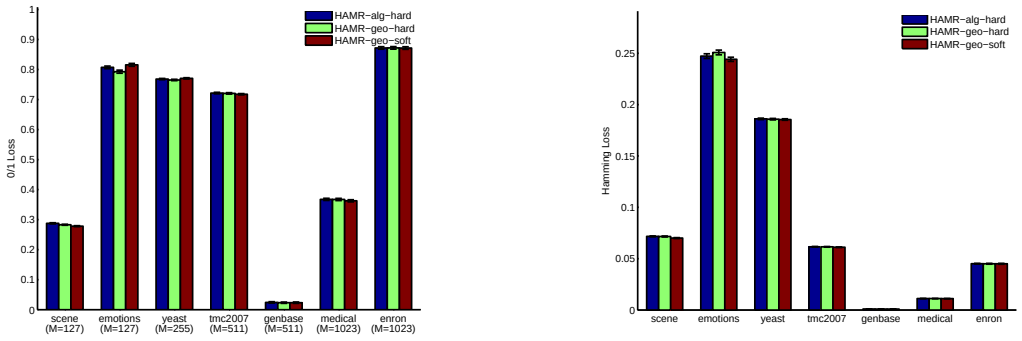
TABLE IV
HAMMING LOSS OF ML-ECC USING BR BASE LEARNERS

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	RREP (BR)	$.0858 \pm .0005$	$.1809 \pm .0018$	$.1903 \pm .0006$	$.0662 \pm .0003$	$.0013 \pm .0001$	$.0183 \pm .0001$	$\mathbf{.0474 \pm .0003}$
	HAMR	$.0726 \pm .0005$	$\mathbf{.1781 \pm .0017}$	$\mathbf{.1878 \pm .0007}$	$.0652 \pm .0002$	$.0012 \pm .0001$	$.0179 \pm .0001$	$\mathbf{.0474 \pm .0003}$
	BCH	$\mathbf{.0717 \pm .0006}$	$.1826 \pm .0018$	$.1898 \pm .0008$	$\mathbf{.0638 \pm .0003}$	$\mathbf{.0010 \pm .0001}$	$\mathbf{.0152 \pm .0001}$	$.0494 \pm .0004$
	LDPC	$.0832 \pm .0006$	$.1877 \pm .0020$	$.1963 \pm .0006$	$.0721 \pm .0003$	$.0014 \pm .0001$	$.0203 \pm .0001$	$.0529 \pm .0004$
Gaussian SVM	RREP (BR)	$.0743 \pm .0005$	$\mathbf{.2461 \pm .0020}$	$\mathbf{.1866 \pm .0006}$	$.0621 \pm .0003$	$.0012 \pm .0001$	$.0113 \pm .0001$	$\mathbf{.0450 \pm .0004}$
	HAMR	$\mathbf{.0717 \pm .0004}$	$\mathbf{.2472 \pm .0023}$	$\mathbf{.1861 \pm .0007}$	$\mathbf{.0616 \pm .0003}$	$\mathbf{.0010 \pm .0001}$	$\mathbf{.0112 \pm .0001}$	$\mathbf{.0451 \pm .0004}$
	BCH	$.0739 \pm .0006$	$.2569 \pm .0030$	$.1880 \pm .0007$	$.0619 \pm .0003$	$\mathbf{.0010 \pm .0001}$	$.0117 \pm .0001$	$.0487 \pm .0005$
	LDPC	$.0742 \pm .0005$	$.2538 \pm .0019$	$.1908 \pm .0006$	$.0688 \pm .0003$	$.0011 \pm .0001$	$.0153 \pm .0001$	$.0517 \pm .0005$
Logistic Regression	RREP (BR)	$.1024 \pm .0006$	$\mathbf{.2062 \pm .0018}$	$\mathbf{.2000 \pm .0007}$	$.0641 \pm .0003$	$.0347 \pm .0007$	$.0212 \pm .0014$	$\mathbf{.0455 \pm .0003}$
	HAMR	$\mathbf{.0959 \pm .0006}$	$\mathbf{.2047 \pm .0022}$	$\mathbf{.2003 \pm .0007}$	$\mathbf{.0635 \pm .0003}$	$.0186 \pm .0005$	$.0191 \pm .0011$	$\mathbf{.0452 \pm .0003}$
	BCH	$\mathbf{.0955 \pm .0006}$	$.2172 \pm .0023$	$.2037 \pm .0008$	$\mathbf{.0638 \pm .0003}$	$\mathbf{.0024 \pm .0002}$	$\mathbf{.0161 \pm .0006}$	$.0472 \pm .0004$
	LDPC	$.1038 \pm .0006$	$.2133 \pm .0019$	$.2044 \pm .0008$	$.0705 \pm .0004$	$.0050 \pm .0004$	$.0234 \pm .0011$	$.0517 \pm .0004$



(a) (b)

Fig. 8. Performance of ML-ECC with hard-/soft-input geometric decoders and algebraic decoder on the BCH code using BR and SVM: (a) $0/1$ loss (b) Hamming loss



(a) (b)

Fig. 9. Performance of ML-ECC with hard-/soft-input geometric decoders and algebraic decoder on the HAMR code using BR and SVM: (a) $0/1$ loss (b) Hamming loss

TABLE V
$0/1$ LOSS OF ML-ECC WITH HARD-/SOFT-INPUT GEOMETRIC DECODERS AND BR

base learner	ECC	decoder	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	HAMR	alg-hard	$.3212 \pm .0021$	$.6574 \pm .0050$	$.7910 \pm .0020$	$\mathbf{.7578 \pm .0025}$	$\mathbf{.0288 \pm .0019}$	$\mathbf{.6387 \pm .0025}$	$\mathbf{.8851 \pm .0036}$
	HAMR	geo-hard	$\mathbf{.3111 \pm .0021}$	$\mathbf{.6480 \pm .0051}$	$\mathbf{.7833 \pm .0022}$	$\mathbf{.7566 \pm .0026}$	$\mathbf{.0280 \pm .0019}$	$\mathbf{.6377 \pm .0027}$	$\mathbf{.8845 \pm .0036}$
	HAMR	geo-soft	$.3294 \pm .0021$	$.6584 \pm .0053$	$.7976 \pm .0020$	$\mathbf{.7588 \pm .0025}$	$\mathbf{.0271 \pm .0017}$	$\mathbf{.6373 \pm .0027}$	$\mathbf{.8848 \pm .0036}$
Gaussian SVM	HAMR	alg-hard	$.2876 \pm .0018$	$.8073 \pm .0043$	$.7681 \pm .0022$	$.7215 \pm .0025$	$\mathbf{.0243 \pm .0022}$	$.3675 \pm .0036$	$\mathbf{.8718 \pm .0042}$
	HAMR	geo-hard	$.2829 \pm .0017$	$\mathbf{.7927 \pm .0051}$	$\mathbf{.7650 \pm .0023}$	$.7205 \pm .0024$	$\mathbf{.0231 \pm .0021}$	$.3671 \pm .0036$	$\mathbf{.8720 \pm .0042}$
	HAMR	geo-soft	$\mathbf{.2782 \pm .0016}$	$.8155 \pm .0048$	$.7705 \pm .0021$	$\mathbf{.7176 \pm .0022}$	$\mathbf{.0233 \pm .0022}$	$\mathbf{.3627 \pm .0035}$	$\mathbf{.8716 \pm .0043}$
Logistic Regression	HAMR	alg-hard	$.4050 \pm .0020$	$.7175 \pm .0054$	$\mathbf{.8282 \pm .0015}$	$.7405 \pm .0024$	$.3509 \pm .0089$	$\mathbf{.5499 \pm .0247}$	$\mathbf{.8740 \pm .0036}$
	HAMR	geo-hard	$.3969 \pm .0022$	$\mathbf{.7097 \pm .0059}$	$\mathbf{.8270 \pm .0016}$	$\mathbf{.7399 \pm .0024}$	$.3541 \pm .0099$	$\mathbf{.5514 \pm .0249}$	$\mathbf{.8744 \pm .0036}$
	HAMR	geo-soft	$\mathbf{.3875 \pm .0024}$	$.7177 \pm .0069$	$\mathbf{.8284 \pm .0016}$	$\mathbf{.7379 \pm .0024}$	$\mathbf{.2777 \pm .0107}$	$\mathbf{.5301 \pm .0229}$	$\mathbf{.8723 \pm .0036}$
Random Forest	BCH	alg-hard	$.2562 \pm .0020$	$.6404 \pm .0060$	$.7792 \pm .0019$	$.7149 \pm .0022$	$\mathbf{.0250 \pm .0019}$	$.4567 \pm .0034$	$.8737 \pm .0038$
	BCH	geo-hard	$\mathbf{.2462 \pm .0019}$	$\mathbf{.6304 \pm .0049}$	$\mathbf{.7217 \pm .0022}$	$\mathbf{.6917 \pm .0020}$	$\mathbf{.0255 \pm .0018}$	$\mathbf{.4130 \pm .0037}$	$.8429 \pm .0038$
	BCH	geo-soft	$.2526 \pm .0020$	$\mathbf{.6264 \pm .0049}$	$.7287 \pm .0022$	$.6949 \pm .0024$	$\mathbf{.0250 \pm .0018}$	$\mathbf{.4157 \pm .0037}$	$\mathbf{.8371 \pm .0043}$
Gaussian SVM	BCH	alg-hard	$.2552 \pm .0018$	$.7809 \pm .0050$	$.7515 \pm .0015$	$.7053 \pm .0024$	$\mathbf{.0255 \pm .0019}$	$.3499 \pm .0030$	$.8561 \pm .0043$
	BCH	geo-hard	$\mathbf{.2503 \pm .0018}$	$\mathbf{.7371 \pm .0051}$	$\mathbf{.7203 \pm .0018}$	$\mathbf{.6975 \pm .0025}$	$\mathbf{.0255 \pm .0019}$	$.3431 \pm .0034$	$.8428 \pm .0045$
	BCH	geo-soft	$\mathbf{.2505 \pm .0019}$	$\mathbf{.7350 \pm .0050}$	$\mathbf{.7212 \pm .0021}$	$\mathbf{.6966 \pm .0030}$	$\mathbf{.0253 \pm .0020}$	$\mathbf{.3376 \pm .0035}$	$\mathbf{.8376 \pm .0045}$
Logistic Regression	BCH	alg-hard	$.3291 \pm .0020$	$.6982 \pm .0048$	$.8094 \pm .0020$	$.7205 \pm .0022$	$\mathbf{.0295 \pm .0018}$	$.4022 \pm .0076$	$.8579 \pm .0038$
	BCH	geo-hard	$.3164 \pm .0017$	$.6886 \pm .0046$	$\mathbf{.7698 \pm .0019}$	$\mathbf{.7102 \pm .0021}$	$.0422 \pm .0026$	$\mathbf{.3704 \pm .0048}$	$\mathbf{.8362 \pm .0040}$
	BCH	geo-soft	$\mathbf{.3142 \pm .0016}$	$\mathbf{.6787 \pm .0046}$	$\mathbf{.7713 \pm .0023}$	$\mathbf{.7112 \pm .0025}$	$.0395 \pm .0026$	$\mathbf{.3670 \pm .0046}$	$.8475 \pm .0040$

TABLE VI
HAMMING LOSS OF ML-ECC WITH HARD-/SOFT-INPUT GEOMETRIC DECODERS AND BR

base learner	ECC	decoder	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	HAMR	alg-hard	$.0726 \pm .0005$	$\mathbf{.1781 \pm .0017}$	$\mathbf{.1878 \pm .0007}$	$\mathbf{.0652 \pm .0002}$	$\mathbf{.0012 \pm .0001}$	$\mathbf{.0179 \pm .0001}$	$\mathbf{.0474 \pm .0003}$
	HAMR	geo-hard	$\mathbf{.0718 \pm .0006}$	$\mathbf{.1768 \pm .0017}$	$\mathbf{.1872 \pm .0007}$	$\mathbf{.0651 \pm .0003}$	$\mathbf{.0012 \pm .0001}$	$\mathbf{.0179 \pm .0001}$	$\mathbf{.0474 \pm .0003}$
	HAMR	geo-soft	$.0726 \pm .0005$	$\mathbf{.1763 \pm .0018}$	$\mathbf{.1873 \pm .0007}$	$\mathbf{.0651 \pm .0002}$	$\mathbf{.0011 \pm .0001}$	$\mathbf{.0179 \pm .0001}$	$\mathbf{.0474 \pm .0003}$
Gaussian SVM	HAMR	alg-hard	$.0717 \pm .0004$	$.2472 \pm .0023$	$\mathbf{.1861 \pm .0007}$	$.0616 \pm .0003$	$\mathbf{.0010 \pm .0001}$	$\mathbf{.0112 \pm .0001}$	$\mathbf{.0451 \pm .0004}$
	HAMR	geo-hard	$.0716 \pm .0005$	$.2508 \pm .0023$	$\mathbf{.1858 \pm .0007}$	$.0615 \pm .0003$	$\mathbf{.0009 \pm .0001}$	$\mathbf{.0112 \pm .0001}$	$\mathbf{.0450 \pm .0004}$
	HAMR	geo-soft	$\mathbf{.0701 \pm .0004}$	$\mathbf{.2441 \pm .0020}$	$\mathbf{.1855 \pm .0007}$	$\mathbf{.0611 \pm .0003}$	$\mathbf{.0010 \pm .0001}$	$\mathbf{.0111 \pm .0001}$	$\mathbf{.0450 \pm .0004}$
Logistic Regression	HAMR	alg-hard	$.0959 \pm .0006$	$\mathbf{.2047 \pm .0022}$	$.2003 \pm .0007$	$.0635 \pm .0003$	$.0186 \pm .0005$	$.0191 \pm .0011$	$\mathbf{.0452 \pm .0003}$
	HAMR	geo-hard	$.0956 \pm .0006$	$.2065 \pm .0023$	$.2002 \pm .0007$	$.0634 \pm .0003$	$.0187 \pm .0006$	$.0192 \pm .0011$	$\mathbf{.0453 \pm .0003}$
	HAMR	geo-soft	$\mathbf{.0920 \pm .0006}$	$\mathbf{.2032 \pm .0024}$	$\mathbf{.1990 \pm .0007}$	$\mathbf{.0631 \pm .0003}$	$\mathbf{.0134 \pm .0005}$	$\mathbf{.0180 \pm .0010}$	$\mathbf{.0453 \pm .0003}$
Random Forest	BCH	alg-hard	$.0717 \pm .0006$	$\mathbf{.1826 \pm .0018}$	$\mathbf{.1898 \pm .0008}$	$.0638 \pm .0003$	$\mathbf{.0010 \pm .0001}$	$.0152 \pm .0001$	$\mathbf{.0494 \pm .0004}$
	BCH	geo-hard	$.0728 \pm .0006$	$.1895 \pm .0017$	$.1968 \pm .0010$	$.0643 \pm .0003$	$\mathbf{.0010 \pm .0001}$	$.0154 \pm .0001$	$.0566 \pm .0005$
	BCH	geo-soft	$\mathbf{.0703 \pm .0006}$	$\mathbf{.1822 \pm .0016}$	$.1910 \pm .0008$	$\mathbf{.0622 \pm .0003}$	$\mathbf{.0010 \pm .0001}$	$\mathbf{.0150 \pm .0002}$	$.0535 \pm .0004$
Gaussian SVM	BCH	alg-hard	$.0739 \pm .0006$	$\mathbf{.2569 \pm .0030}$	$\mathbf{.1880 \pm .0007}$	$\mathbf{.0619 \pm .0003}$	$\mathbf{.0010 \pm .0001}$	$\mathbf{.0117 \pm .0001}$	$\mathbf{.0487 \pm .0005}$
	BCH	geo-hard	$.0735 \pm .0005$	$.2761 \pm .0031$	$.1952 \pm .0007$	$.0649 \pm .0003$	$\mathbf{.0010 \pm .0001}$	$.0126 \pm .0001$	$.0577 \pm .0006$
	BCH	geo-soft	$\mathbf{.0721 \pm .0006}$	$.2614 \pm .0028$	$.1911 \pm .0007$	$.0624 \pm .0003$	$\mathbf{.0011 \pm .0001}$	$.0121 \pm .0001$	$.0534 \pm .0005$
Logistic Regression	BCH	alg-hard	$.0955 \pm .0006$	$\mathbf{.2172 \pm .0023}$	$\mathbf{.2037 \pm .0008}$	$\mathbf{.0638 \pm .0003}$	$\mathbf{.0024 \pm .0002}$	$.0161 \pm .0006$	$\mathbf{.0472 \pm .0004}$
	BCH	geo-hard	$.0957 \pm .0006$	$.2312 \pm .0027$	$.2116 \pm .0007$	$.0673 \pm .0003$	$.0054 \pm .0004$	$.0154 \pm .0004$	$.0558 \pm .0004$
	BCH	geo-soft	$\mathbf{.0913 \pm .0006}$	$\mathbf{.2170 \pm .0026}$	$.2067 \pm .0008$	$\mathbf{.0640 \pm .0003}$	$.0046 \pm .0003$	$\mathbf{.0141 \pm .0003}$	$.0526 \pm .0004$

The experiment setting is basically the same, but we only use `scene` and `emotions` datasets, with Random Forests or logistic regression base learners. Both real-valued ECC methods limit their codeword length to at most twice of the number of labels, and the codeword contains K binary bits for original labels and at most K real-valued bits. In the following experiment, we take all $2K$ binary and real-valued bits for the real-valued ECC methods. There is a parameter λ in decoding for balancing the two parts, and we set it to 1 (equally weighted). For our ML-ECC framework, we consider HAMR and BCH code with the proposed soft-input geometric decoder, and use 127-bit binary codewords. Besides the $0/1$ loss and the Hamming loss, we also report micro and macro F_1 score following [16] and [17].

The results on Random Forests learners are shown in Table VII, and the results on logistic regression learners are shown in Table VIII. It can be seen that with a stronger base learner like Random Forests, the HAMR and BCH codes are better than both real-valued ECC methods on the two datasets and on most of the measures. With the logistic regression learner, while BCH code performs the best on `scene` dataset, it only wins on $0/1$ loss on `emotions` dataset. The real-valued ECC methods give higher micro and macro F_1 score than HAMR and BCH on the `emotions` dataset. The reason may be that the power of logistic regression base learner is limited, and the parity bits of HAMR and BCH are too difficult for the base learner. On the other hand, the code generated by CCA-OC and MaxMargin methods are easier to learn for such a linear model. The results demonstrate that the proposed binary-ECC-based framework coupled with a sufficiently sophisticated base learner is a better choice for multi-label classification with ECC.

V. CONCLUSION

We presented a framework for applying the ECCs on multi-label classification. We then studied the use of four classic ECC designs, namely the RREP, HAMR, BCH, and LDPC. We showed that RREP can be used to give a new perspective of the RAKEL algorithm as a special instance of the framework with the k-powerset as the base learner.

TABLE VII
COMPARISON BETWEEN ML-ECC AND REAL-VALUED ECC METHODS USING RANDOM FORESTS BASE LEARNERS

		scene		
ECC	$0/1$ loss	Hamming loss	Micro-F_1	Macro-F_1
HAMR-geo-soft	$.329 \pm .002$	$.073 \pm .001$	$.771 \pm .002$	$.774 \pm .002$
BCH-geo-soft	$\mathbf{.253 \pm .002}$	$\mathbf{.070 \pm .001}$	$\mathbf{.794 \pm .002}$	$\mathbf{.801 \pm .002}$
CCA-OC	$.317 \pm .001$	$.093 \pm .000$	$.732 \pm .001$	$.740 \pm .001$
		emotions		
ECC	$0/1$ loss	Hamming loss	Micro-F_1	Macro-F_1
HAMR-geo-soft	$.658 \pm .005$	$\mathbf{.176 \pm .002}$	$.702 \pm .003$	$.676 \pm .003$
BCH-geo-soft	$\mathbf{.626 \pm .005}$	$.182 \pm .002$	$\mathbf{.715 \pm .003}$	$\mathbf{.698 \pm .003}$
CCA-OC	$.673 \pm .002$	$.202 \pm .001$	$.692 \pm .001$	$.682 \pm .001$

TABLE VIII
COMPARISON BETWEEN ML-ECC AND REAL-VALUED ECC METHODS USING LOGISTIC REGRESSION BASE LEARNERS

		scene		
ECC	$0/1$ loss	Hamming loss	Micro-F_1	Macro-F_1
HAMR-geo-soft	$.388 \pm .002$	$.092 \pm .001$	$.716 \pm .002$	$.720 \pm .002$
BCH-geo-soft	$\mathbf{.314 \pm .002}$	$\mathbf{.091 \pm .001}$	$\mathbf{.734 \pm .002}$	$\mathbf{.740 \pm .001}$
CCA-OC	$.360 \pm .001$	$.109 \pm .000$	$.688 \pm .001$	$.695 \pm .001$
MaxMargin	$.365 \pm .002$	$.111 \pm .001$	$.682 \pm .002$	$.689 \pm .003$
		emotions		
ECC	$0/1$ loss	Hamming loss	Micro-F_1	Macro-F_1
HAMR-geo-soft	$.718 \pm .007$	$\mathbf{.203 \pm .002}$	$.654 \pm .005$	$.631 \pm .005$
BCH-geo-soft	$\mathbf{.679 \pm .005}$	$.217 \pm .003$	$.665 \pm .004$	$.650 \pm .004$
CCA-OC	$.681 \pm .002$	$.207 \pm .001$	$\mathbf{.679 \pm .001}$	$\mathbf{.672 \pm .001}$
MaxMargin	$.686 \pm .003$	$.210 \pm .001$	$.677 \pm .001$	$.668 \pm .001$

We conducted experiments with the four ECC designs on various real-world datasets. The experiments further clarified the trade-off between the strength of the ECC and the hardness of the base learning tasks. Experimental results demonstrated that several ECC designs can lead to a better use of the trade-off. For instance, HAMR is superior over RREP for the k-powerset base learners because it leads to a new algorithm that is better than the original RAKEL in terms of $0/1$ loss while maintaining a comparable level of Hamming loss; BCH is another superior design, which could significantly improve RAKEL in terms of $0/1$ loss. When compared with the traditional BR algorithm, we showed that using a stronger ECC like HAMR or BCH can lead to better performance in terms of both $0/1$ and Hamming loss.

The results justify the validity and usefulness of the framework when coupled with some classic ECC. An interesting future direction is to consider adaptive ECC like the ones studied for multi-class classification [5], [6].

Besides the framework, we also presented a novel geometric decoder for general linear code based on approximating the XOR operation by multiplication. This decoder is capable of not only taking hard input as algebraic decoders, but also taking soft input from the channel into account. The soft input may be gathered from the base learner channels as their confidence of an instance to be in one class. The experimental result on this new decoder demonstrated that this decoder outshines the ordinary decoder in terms of $0/1$ loss and using soft input from the Binary Relevance learner further improves the performance of this decoder on Hamming loss. It remains an interesting research problem on appropriately gathering soft input from the k-powerset learner.

REFERENCES

[1] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. 9th Int. Conf. Music Inform. Retrieval*, 2008.

[2] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, 2004.

[3] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, 1948.

[4] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artificial Intell. Research*, vol. 2, 1995.

[5] R. E. Schapire, "Using output codes to boost multiclass learning problems," in *Proc. 14th Int. Conf. Mach. Learning*, 1997.

[6] L. Li, "Multiclass boosting with repartitioning," in *Proc. 23rd Int. Conf. Mach. Learning*, 2006.

[7] A. Z. Kouzani and G. Nasireding, "Multilabel classification by BCH code and random forests," *Int. J. Recent Trends Eng.*, vol. 2, no. 1, 2009.

[8] A. Z. Kouzani, "Multilabel classification using error correction codes," in *Proc. 5th Int. Conf. Advances Computation Intell.*, 2010.

[9] C.-S. Ferng and H.-T. Lin, "Multi-label classification with error-correcting codes," in *Proc. 3rd Asian Conf. Mach. Learning*, 2011.

[10] C.-S. Ferng, "Multi-label classification with hard-/soft-decoded error-correcting codes," Master's thesis, National Taiwan University, 2012.

[11] F. Tai and H.-T. Lin, "Multi-label classification with principal label space transformation," *Neural Computation*, 2012.

[12] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence in multi-label classification," in *Proc. 2nd Int. Workshop Learning Multi-Label Data*, 2010.

[13] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Proc. 18th European Conf. Mach. Learning*, 2007.

[14] D. Hsu, S. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing," in *Proc. 24th Neural Inform. Process. Syst.*, 2009.

[15] D. J. C. Mackay, *Information Theory, Inference and Learning Algorithms*, 1st ed. Cambridge Univ. Press, 2003.

[16] Y. Zhang and J. Schneider, "Multi-label output codes using canonical correlation analysis," in *Proc. 14th Int. Conf. Artificial Intell. Statistics*, 2011.

[17] ——, "Maximum margin output coding," in *Proc. 29th Int. Conf. Mach. Learning*, 2012.

[18] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Tech. J.*, vol. 26, no. 2, 1950.

[19] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error correcting binary group codes," *Information and Control*, vol. 3, no. 1, 1960.

[20] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, 1959.

[21] R. G. Gallager, *Low Density Parity Check Codes, Monograph*. MIT Press, 1963.

[22] G. Tsoumakas, J. Vilcek, and E. S. Xioufis, "MULAN: A Java library for multi-label learning," 2010.

[23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[24] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[25] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learning Research*, vol. 9, 2008.