國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

剛性與柔性解碼之錯誤更正碼於多標籤分類學習之應用

Multi-label Classification with Hard-/soft-decoded Error-correcting Codes

馮俊菘

Ferng, Chun-Sung

指導教授：林軒田 博士

Advisor: Hsuan-Tien Lin, Ph.D.

中華民國 101 年 6 月

June, 2012

致謝

能寫出這份論文和得到碩士學位，首先我要感謝林軒田老師。 謝謝老師一直以來的指導，每個星期都和我一對一的討論。 從一開始領我進門，與我討論題目、想法和實驗，到最後論文的修改、簡報的技巧， 都是在老師的教導和指引下才能有如此成果。 還有要感謝口試委員李育杰老師和林守德老師，不但百忙之中撥空來參與口試， 也給了我不少有用的建議，讓這篇論文更加完整。

還有，要感謝 217 和 536 實驗室的同學們，在這兩年願意不時和我討論，讓我對機器學習以及其他相關領域有更多認識； 在我準備口頭報告和口試的時候願意聽我練習，讓我能表現得更好。 此外，也要感謝我的女朋友琇文，一路上陪著我一起努力，在我遭遇困難的時候給我安慰、幫我加油打氣，讓我可以繼續前進。

最重要的，要感謝爸爸媽媽，因為有你們的付出和支持，我才能心無旁騖的在學業上努力。

在此對所有幫助我的人至上最誠摯的感謝，謝謝你們。

馮俊菘， 2012 年 7 月

中文摘要

我們提出一個將錯誤更正碼 (error-correcting codes, ECC) 應用於多標籤分類問題 (multi-label classification) 的架構。 在這個架構中，我們以一些基礎學習器 (base learner) 當作有干擾的傳輸頻道， 並用錯誤更正碼來更正這些基礎學習器的預測錯誤。 透過這個架構，我們可以用簡單的重複碼 (repetition code) 來解釋現有的隨機 k 標籤組演算法 (random k-label-sets, RAKEL) 。 我們也實驗了各種錯誤更正碼應用在多標籤分類問題的效果， 實驗結果顯示，利用較強的錯誤更正碼可以改善隨機 k 標籤組演算法的表現； 此外，讓傳統的二元關聯演算法 (binary relevance) 學習一些校驗標籤 (parity-checking labels) 也會讓它有更好的表現。 而且，由不同的錯誤更正碼的實驗結果可以看出，錯誤更正碼的強度會影響基礎學習器的難度，妥善平衡兩者可以讓結果變得更好。 最後，我們也設計了一個新的解碼器來處理剛性（二元值）與柔性（實數值）的線性錯誤更正碼， 實驗結果也證實這個新的解碼器可以提昇這個架構的表現。

關鍵詞：機器學習、多標籤分類、錯誤更正碼、柔性解碼、幾何解碼

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 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 linear ECC for either hard (binary) or soft (real-valued) bits, and design a novel decoder for the ECC. We demonstrate that the decoder improves the performance of our framework.

Keywords: Machine Learning, Multi-label Classification, Error-correcting Codes, Soft Decoding, Geometric Decoding.

Contents

List of Figures

List of Tables

Chapter 1

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 text and music categorization [Pestian et al., 2007, Trohidis et al., 2008], scene analysis [Boutell et al., 2004], and genomics [Elisseeff and Weston, 2002, Diplaris et al., 2005], multi-label classification is attracting much research attention in recent years.

Error-correcting code (ECC) roots from the information theoretic pursuit of communication [Shannon, 1948]. 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 [Dietterich and Bakiri, 1995]. In particular, with the help of the ECC, we can reduce multi-class classification to several binary classification tasks. Then, following the foundation of the ECC in information theory [Shannon, 1948, Mackay, 2003], a suitable ECC can correct a small portion of binary classification errors during the prediction stage and thus improve the prediction accuracy. Several designs, including some classic ECC [Dietterich and Bakiri, 1995] and some adaptively constructed ECC [Schapire, 1997, Li, 2006], 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. Kouzani and Nasireding [2009] take the first step in this direction by proposing a multi-label classification approach that applies a classic ECC, the Bose-Chaudhuri-Hocquenghem (BCH) code, using a batch of binary classifiers as the noisy channel. The work is followed by some extensions to the convolution code [Kouzani, 2010]. 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 [Dietterich and Bakiri, 1995] and for multi-label classification [Kouzani and Nasireding, 2009]. 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 either hard or soft inputs.

The thesis is organized as follows. First, we introduce the multi-label classification problem in Section 1.1, and present related works in Section 1.2. Chapter 2 illustrates the framework and demonstrates its effectiveness. Chapter 3 presents a new decoder for hard or soft inputs. Finally we conclude in Chapter 4.

1.1 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 Tai and Lin [2012], 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 $h(\mathbf{x})$ predicts $\mathbf{y}$ well. Two common ones are:

• **subset** $0/1$ **loss**: this loss function is arguably one of the most challenging loss functions because zero (small) loss occurs only when every bit of the prediction is correct.

$$\Delta_{0/1}(\tilde{\mathbf{y}}, \mathbf{y}) = [\![\tilde{\mathbf{y}} \neq \mathbf{y}]\!]$$

• **Hamming loss**: this loss function considers individual bit differences.

$$\Delta_{HL}(\tilde{\mathbf{y}}, \mathbf{y}) = \frac{1}{K} \sum_{i=1}^{K} [\![\tilde{\mathbf{y}}[i] \neq \mathbf{y}[i]]\!]$$

Dembczyński et al. [2010] 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 [Tsoumakas and Vlahavas, 2007], in this paper we only focus on $0/1$ and Hamming because they connect tightly with the ECC framework that will be discussed.[1]

1.2 Related Works

The hypercube view [Tai and Lin, 2012] unifies many existing problem transformation approaches [Tsoumakas and Vlahavas, 2007] for multi-label classification. Problem transformation approaches transform multi-label classification into one or more reduced learning tasks. For instance, one simple problem transformation approach for multi-label classification 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) [Tsoumakas and Vlahavas, 2007], which transforms multi-label classification into many multi-class classification tasks with a smaller number of extended labels.

Multi-label classification with compressive sensing [Hsu et al., 2009] is a problem transformation approach that encodes the training label set $\mathbf{y}_n$ to a shorter, real-valued codeword vector using compressive sensing. Tai and Lin [2012] study some different encoding schemes from label sets to real-valued codewords. Note that those encoding schemes focus on *compression*—removing the redundancy within the binary signals (label sets) to form shorter codewords. The compression perspective can lead to not only more efficient training and testing but also more meaningful codewords.

Compression is a classic task in information theory based on Shannon's first theorem [Shannon, 1948]. 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 theo-

[1]We follow the final remark of Dembczyński et al. [2010] to only focus on the loss functions that are related to our algorithmic goals.

rem [Shannon, 1948]. The 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 [Mackay, 2003], 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 the next chapter.

Chapter 2

ECC for Multi-label Classification

2.1 ML-ECC Framework

We now describe the 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 in Algorithm 1.

Algorithm 1 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 [Kouzani and Nasireding, 2009] uses the BCH code (see Subsection 2.2.3) as the ECC and BR on Random Forest as the base learner $\mathcal{A}_b$. Note that Kouzani and Nasireding [2009] 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 $\Delta_{0/1}$.

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 2.2. Then, if Δ_{HL} of $\tilde{h}$ is

Algorithm 1 Error-Correcting Framework

```
• 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.
```

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*

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

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

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

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 [Li, 2006]. In the next section, we study the ECC with different strength and empirically verify the trade-off in Section 2.4.

2.2 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: the repetition code, the Hamming on repetition code, the Bose-Chaudhuri-Hocquenghem code, and the low-density parity-check code.

2.2.1 Repetition Code

One of the simplest ECCs is repetition code (REP) [Mackay, 2003], for which every bit in $\mathbf{y}$ is repeated $\lfloor \frac{M}{K} \rfloor$ times in $\mathbf{b}$ during encoding. If M is not a multiple of K, then $(M \bmod K)$ bits are repeated one more time. The decoding takes a majority vote using the received copies of each bit. Because of the majority vote, repetition code corrects up to $m_{REP} = \frac{1}{2}\lfloor \frac{M}{K} \rfloor - 1$ bit errors in $\mathbf{b}$. We will discuss the connection between REP and the RAKEL algorithm in Section 2.3.

2.2.2 Hamming on Repetition Code

A slightly more complicated ECC than REP is called the Hamming code (HAM) [Hamming, 1950], 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 $m_{REP} = \frac{1}{2}\lfloor \frac{M}{K} \rfloor - 1$ 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 and K is a multiple of 4, it can be proved that $m_{HAMR} = \frac{4M}{7K}$, which is generally better than $m_{REP} = \frac{1}{2}\lfloor \frac{M}{K} \rfloor - 1$. 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.

2.2.3 Bose-Chaudhuri-Hocquenghem Code

BCH was invented by Bose and Ray-Chaudhuri [1960], and independently by Hocquenghem [1959]. It 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 [Mackay, 2003], 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.

2.2.4 Low-density Parity-check Code

Low-density parity-check code (LDPC) [Mackay, 2003] is recently drawing much research attention in communications. LDPC shares an interesting connection between ECC and Bayesian learning [Mackay, 2003]. 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 [Gallager, 1963], 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.

2.3 ECC View of RAKEL

RAKEL is a multi-label classification algorithm proposed by Tsoumakas and Vlahavas [2007]. 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 $\mathcal{A}_b$. Next, we empirically study how the two parts affect the performance of multi-label classification.

2.4 Experimental Results

We compare RREP, HAMR, BCH, and LDPC with the ECC framework on seven real-world datasets in different domains: `scene`, `emotions`, `yeast`, `tmc2007`, `genbase`, `medical`, and `enron` [Tsoumakas et al., 2010]. The statistics of these datasets are shown in Table 2.1. 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 has 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

Table 2.1: Dataset characteristics

DATASET	K	TRAINING	TESTING	FEATURES
SCENE	6	1211	1196	294
EMOTIONS	6	391	202	72
YEAST	14	1500	917	103
TMC2007	22	1430	1430	500
GENBASE	27	463	199	1186
MEDICAL	45	333	645	1449
ENRON	53	1123	579	1001

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 2.4.4.

We take the default Random Forest from Weka [Hall et al., 2009] with 60 trees. For the non-linear SVM, we use LIBSVM [Chang and Lin, 2001] 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 [Fan et al., 2008] for the logistic regression and choose the parameter C by cross validation from $\{2^{-5}, 2^{-3}, \cdots, 2^{7}\}$.

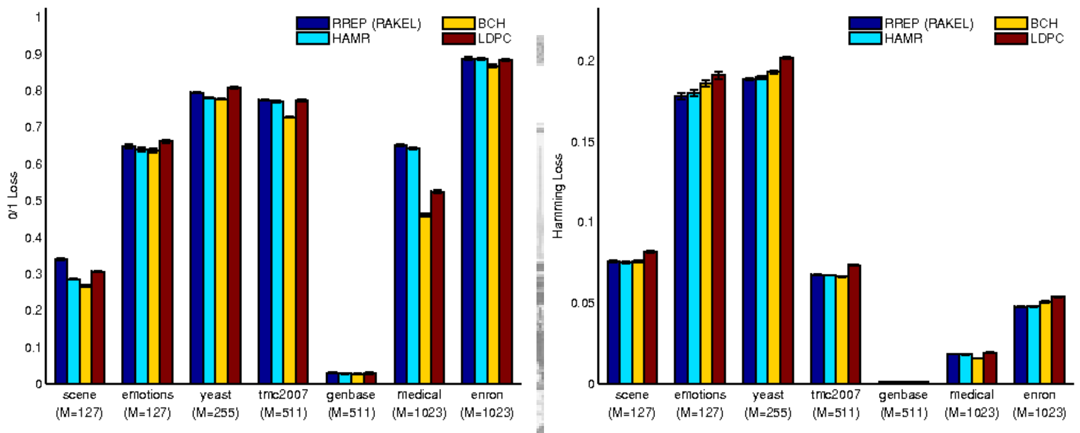
Note that the experiments taken in this work are generally broader than existing works that are related to multi-label classification with the ECC in terms of the datasets, the codes, the "channels," and the base learners, as shown in Table 2.2. The goal of the experiments is not only to justify that the framework is promising but also to rigorously identify the best codes, channels, and base learners for solving general multi-label classification tasks via the ECC.

2.4.1 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

Table 2.2: Focus of existing works under the ML-ECC framework

work	# datasets	codes	channels	base learners
RAKEL [Tsoumakas and Vlahavas, 2007]	3	RREP	k-powerset	linear SVM
ML-BCHRF [Kouzani and Nasireding, 2009]	3	BCH	BR	Random Forest
ML-BCHRF & ML-CRF [Kouzani, 2010]	1	convolution, and BCH	BR	Random Forest
this work	7	RREP, HAMR, BCH, and LDPC	3-powerset, and BR	Random Forest, Gaussian SVM, logistic regression



(a) $0/1$ loss (b) Hamming loss

Figure 2.1: Performance of ML-ECC using the 3-powerset with Random Forests

experiments on different codeword lengths are presented in Section 2.4.2. Here the base multi-label learner is the 3-powerset with Random Forests. Following the description in Section 2.3, RREP with the 3-powerset is exactly the same as RAKEL with $k = 3$.

The results on $0/1$ loss is shown in Figure 2.1(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

Table 2.3: $0/1$ loss of ML-ECC using 3-powerset base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (RAKEL)	.3390 ± .0022	.6475 ± .0057	.7939 ± .0022	.7738 ± .0025
Random Forest	HAMR	.2855 ± .0022	**.6393 ± .0055**	.7789 ± .0021	.7693 ± .0024
Random Forest	BCH	**.2671 ± .0020**	**.6366 ± .0061**	**.7764 ± .0021**	**.7273 ± .0018**
Random Forest	LDPC	.3058 ± .0024	.6606 ± .0050	.8080 ± .0024	.7728 ± .0022
Gaussian SVM	RREP (RAKEL)	.2856 ± .0016	**.7759 ± .0055**	.7601 ± .0023	.7196 ± .0024
Gaussian SVM	HAMR	.2635 ± .0017	**.7729 ± .0052**	.7530 ± .0021	.7162 ± .0023
Gaussian SVM	BCH	**.2576 ± .0017**	**.7744 ± .0053**	**.7429 ± .0017**	**.7095 ± .0020**
Gaussian SVM	LDPC	.2780 ± .0020	.8040 ± .0044	.7574 ± .0021	.7403 ± .0019
Logistic Regression	RREP (RAKEL)	.3601 ± .0019	**.6949 ± .0070**	.8161 ± .0017	.7408 ± .0024
Logistic Regression	HAMR	.3299 ± .0018	**.6954 ± .0057**	.8061 ± .0019	.7383 ± .0025
Logistic Regression	BCH	**.3148 ± .0018**	.7068 ± .0046	**.7899 ± .0020**	**.7233 ± .0024**
Logistic Regression	LDPC	.3655 ± .0028	.7295 ± .0056	.8082 ± .0024	.7562 ± .0027

base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	RREP (RAKEL)	.0295 ± .0021	.6508 ± .0024	.8866 ± .0038
Random Forest	HAMR	**.0276 ± .0021**	.6420 ± .0029	.8855 ± .0036
Random Forest	BCH	**.0263 ± .0020**	**.4598 ± .0036**	**.8659 ± .0039**
Random Forest	LDPC	.0288 ± .0021	.5238 ± .0032	.8830 ± .0036
Gaussian SVM	RREP (RAKEL)	.0295 ± .0025	.3679 ± .0036	.8725 ± .0041
Gaussian SVM	HAMR	.0303 ± .0026	.3641 ± .0031	.8693 ± .0042
Gaussian SVM	BCH	**.0255 ± .0019**	**.3394 ± .0027**	**.8477 ± .0045**
Gaussian SVM	LDPC	.0285 ± .0021	.3856 ± .0031	.8666 ± .0041
Logistic Regression	RREP (RAKEL)	.3593 ± .0078	.5507 ± .0254	.8762 ± .0035
Logistic Regression	HAMR	.2275 ± .0099	.5268 ± .0230	.8754 ± .0035
Logistic Regression	BCH	**.0250 ± .0018**	**.3797 ± .0044**	**.8504 ± .0042**
Logistic Regression	LDPC	.0325 ± .0018	.4516 ± .0083	.8653 ± .0038

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 2.1(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 2.3 and 2.4. 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

Table 2.4: 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)
Random Forest	RREP (RAKEL)	$.0755 \pm .0006$	$\mathbf{.1778 \pm .0018}$	$\mathbf{.1884 \pm .0007}$	$.0674 \pm .0003$
Random Forest	HAMR	$\mathbf{.0748 \pm .0006}$	$.1798 \pm .0019$	$.1894 \pm .0008$	$.0671 \pm .0003$
Random Forest	BCH	$\mathbf{.0753 \pm .0007}$	$.1858 \pm .0021$	$.1928 \pm .0008$	$\mathbf{.0662 \pm .0003}$
Random Forest	LDPC	$.0817 \pm .0007$	$.1907 \pm .0021$	$.2012 \pm .0007$	$.0734 \pm .0003$
Gaussian SVM	RREP (RAKEL)	$\mathbf{.0719 \pm .0005}$	$\mathbf{.2432 \pm .0021}$	$\mathbf{.1853 \pm .0007}$	$\mathbf{.0613 \pm .0003}$
Gaussian SVM	HAMR	$\mathbf{.0723 \pm .0005}$	$.2492 \pm .0023$	$.1868 \pm .0006$	$\mathbf{.0610 \pm .0003}$
Gaussian SVM	BCH	$.0739 \pm .0006$	$.2644 \pm .0019$	$.1898 \pm .0008$	$.0629 \pm .0003$
Gaussian SVM	LDPC	$.0755 \pm .0006$	$.2634 \pm .0027$	$.1917 \pm .0007$	$.0679 \pm .0003$
Logistic Regression	RREP (RAKEL)	$\mathbf{.0915 \pm .0005}$	$\mathbf{.2026 \pm .0025}$	$\mathbf{.1993 \pm .0007}$	$\mathbf{.0634 \pm .0003}$
Logistic Regression	HAMR	$\mathbf{.0911 \pm .0005}$	$.2064 \pm .0024$	$.2003 \pm .0007$	$\mathbf{.0634 \pm .0003}$
Logistic Regression	BCH	$.0920 \pm .0005$	$.2233 \pm .0022$	$.2051 \pm .0008$	$.0653 \pm .0003$
Logistic Regression	LDPC	$.0989 \pm .0007$	$.2202 \pm .0021$	$.2054 \pm .0007$	$.0701 \pm .0003$
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (RAKEL)	$.0012 \pm .0001$	$.0182 \pm .0001$	$\mathbf{.0477 \pm .0004}$	
Random Forest	HAMR	$.0012 \pm .0001$	$.0180 \pm .0001$	$\mathbf{.0479 \pm .0004}$	
Random Forest	BCH	$\mathbf{.0011 \pm .0001}$	$\mathbf{.0159 \pm .0001}$	$.0506 \pm .0004$	
Random Forest	LDPC	$.0013 \pm .0001$	$.0192 \pm .0002$	$.0538 \pm .0005$	
Gaussian SVM	RREP (RAKEL)	$.0013 \pm .0001$	$\mathbf{.0112 \pm .0001}$	$\mathbf{.0449 \pm .0004}$	
Gaussian SVM	HAMR	$.0013 \pm .0001$	$\mathbf{.0111 \pm .0001}$	$\mathbf{.0449 \pm .0004}$	
Gaussian SVM	BCH	$\mathbf{.0010 \pm .0001}$	$.0114 \pm .0001$	$.0516 \pm .0006$	
Gaussian SVM	LDPC	$.0014 \pm .0001$	$.0140 \pm .0001$	$.0530 \pm .0005$	
Logistic Regression	RREP (RAKEL)	$.0179 \pm .0006$	$.0190 \pm .0011$	$\mathbf{.0453 \pm .0003}$	
Logistic Regression	HAMR	$.0102 \pm .0005$	$.0176 \pm .0009$	$\mathbf{.0454 \pm .0003}$	
Logistic Regression	BCH	$\mathbf{.0013 \pm .0001}$	$\mathbf{.0137 \pm .0003}$	$.0505 \pm .0004$	
Logistic Regression	LDPC	$.0024 \pm .0002$	$.0187 \pm .0006$	$.0528 \pm .0004$	

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}. We also report the micro and macro F_1 scores, and also the pairwise label ranking loss in Tables A.1, A.2, and A.3, respectively.

2.4.2 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.

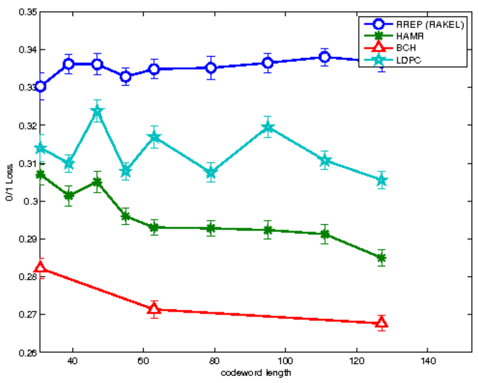
The performance of the ECC framework with different codeword lengths on the `scene` dataset is shown on Figure 2.2. Here, the base learner is again the 3-powerset with Random Forests. The codeword length M varies from 31 to 127, which is about 5 to 20 times of number of labels L. We do not include shorter 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 2.2(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.

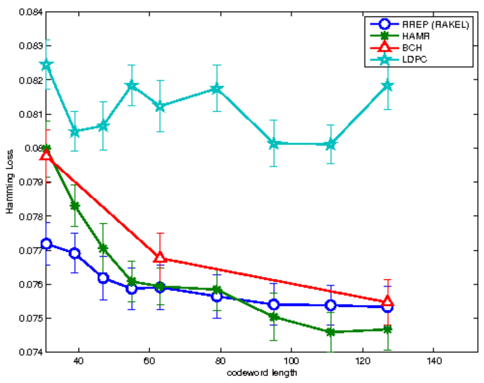
The $\Delta_{0/1}$ of HAMR and BCH are slightly decreasing with M, but the differences between $M = 63$ and $M = 127$ are generally small (smaller than the differences between $M = 31$ and $M = 63$, in particular). This indicates that learning some parity bits provides additional information for prediction, 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 2.2(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}.

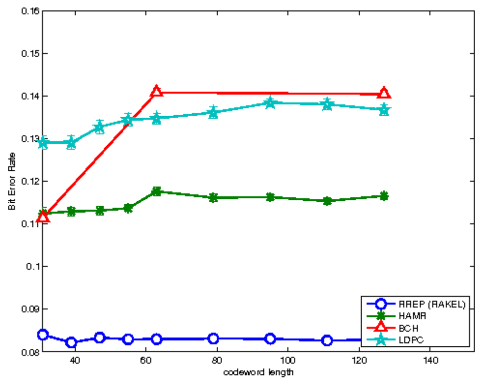
As stated in Sections 1.1 and 2.1, the base learners serve as the channels in the ECC framework and the performance of base learners may be affected by the codes. Therefore, using a strong ECC does not always improve multi-label classification performance. Next, we verify the trade-off by measuring the bit error rate Δ_{BER} of $\tilde{h}$, which is defined as the



(a) 0/1 loss



(b) Hamming loss

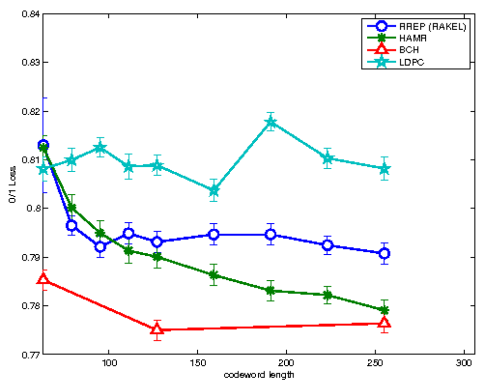


(c) bit-error rate

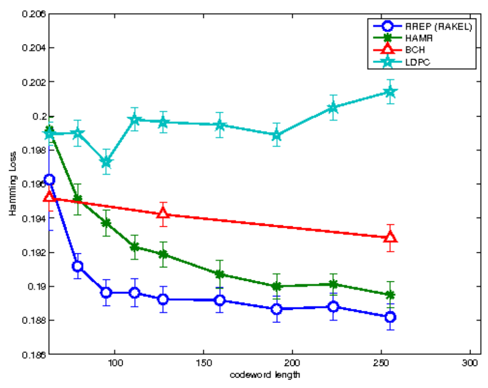
Figure 2.2: Varying codeword length on `scene`: ML-ECC using the 3-powerset with Random Forests

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.

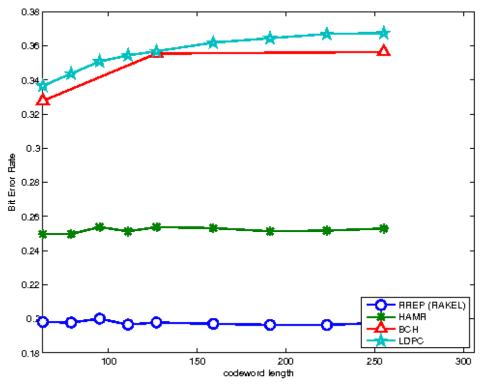
Figure 2.2(c) shows the Δ_{BER} versus M for each ECC. RREP has almost constant bit error rate. HAMR also has nearly constant bit error rate but at a higher value. The bit error rate of BCH is similar to that of HAMR when the codeword is short, but the bit error rate increases with M. One explanation 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 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



(a) 0/1 loss



(b) Hamming loss

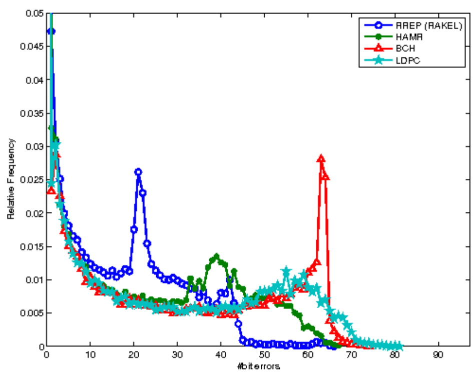


(c) bit-error rate

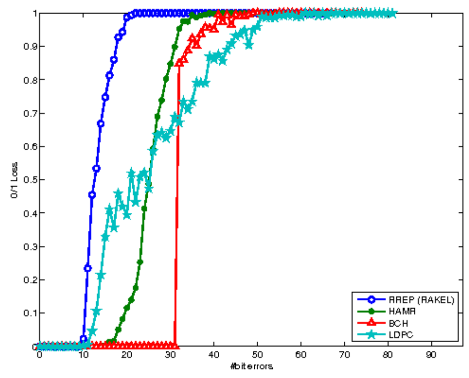
Figure 2.3: Varying codeword length on `yeast`: ML-ECC using the 3-powerset with Random Forests

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.

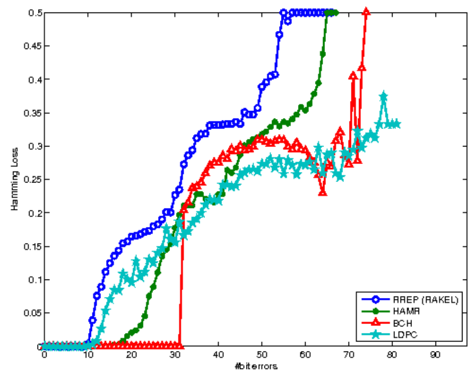
Similar results show up in other datasets with all three base learners. The performance on the `yeast` dataset with the 3-powerset and Random Forests is shown in Figure 2.3. Because the number of labels in the `yeast` dataset is about twice of that in the `scene` dataset, the codeword length here ranges 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.



(a) relative frequency vs. number of bit errors



(b) $0/1$ loss vs. number of bit errors



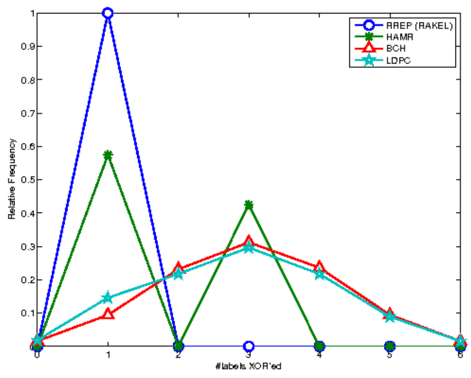
(c) Hamming loss vs. number of bit errors

Figure 2.4: Bit errors and losses: the `scene` dataset, $M = 127$

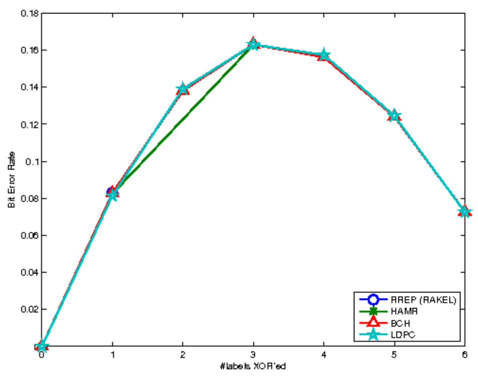
2.4.3 Bit Error Analysis

To further analyze the difference between different ECC designs, we zoom in to $M = 127$ of Figure 2.2. 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 2.4(a). The average $\Delta_{0/1}$ and Δ_{HL} of each group are also plotted in Figure 2.4(b) and 2.4(c). The curve of each ECC forms two peak regions in Figure 2.4(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 2.2(c).

We can clearly see the difference on the strength of different ECC from Figure 2.4(b). BCH can tolerate up to 31-bit errors, but its $\Delta_{0/1}$ sharply increases over 0.8 for 32-bit er-



(a) relative frequency vs. number of labels XOR'ed



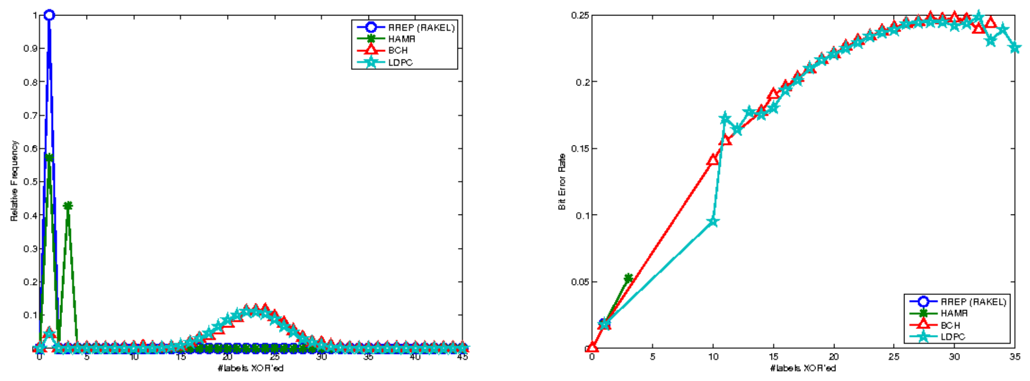
(b) bit error rate vs. number of labels XOR'ed

Figure 2.5: Parity bits: the `scene` dataset, 6 labels, $M = 127$

rors. 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 2.4(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 2.4(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 2.4(a) and the growth in Figure 2.4(c) and thus lower overall Δ_{HL}.

Figure 2.4(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 2.5 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 2.5(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



(a) relative frequency vs. number of labels XOR'ed (b) bit error rate vs. number of labels XOR'ed

Figure 2.6: Parity bits: the `medical` dataset, 45 labels, $M = 1023$

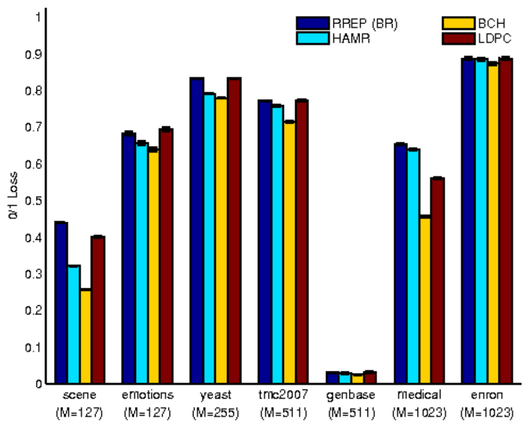
(0) to all (6) labels, while most of the bits are the XOR of half of the labels (3 labels).

Next we show how well the base learners learned on each group in Figure 2.5(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 2.2(c).

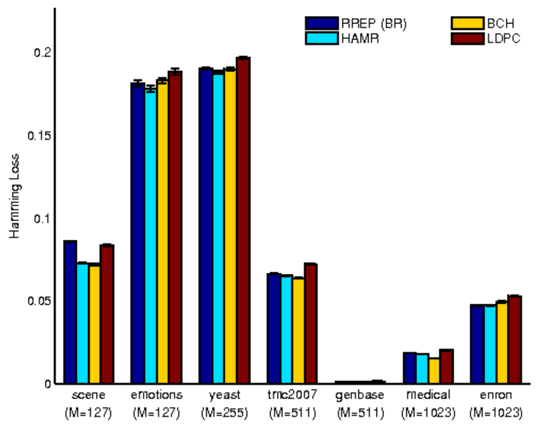
These findings also appear on other datasets and other base learners, such as `medical` dataset (45 labels, $M = 1023$) shown in Figure 2.6. BCH and LDPC have many bits XOR'ed from about half of the labels, and the transformed learning tasks of such bits are harder to learn than that of original labels.

2.4.4 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



(a) $0/1$ loss



(b) Hamming loss

Figure 2.7: Performance of ML-ECC using Binary Relevance with Random Forests

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 2.7(a). From the figure, we can see that BCH and HAMR has 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 2.7(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.

Experiments with other base learners also support similar findings, as shown in Tables 2.5 and 2.6. Notice that HAMR performs better than BCH when using Gaussian SVM base learners. Thus, extending BR by learning some more parity bits and decoding

Table 2.5: $0/1$ loss of ML-ECC using BR base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (BR)	$.4396 \pm .0022$	$.6825 \pm .0053$	$.8332 \pm .0016$	$.7715 \pm .0023$
Random Forest	HAMR	$.3213 \pm .0020$	$.6573 \pm .0051$	$.7910 \pm .0020$	$.7578 \pm .0025$
Random Forest	BCH	$\mathbf{.2570 \pm .0022}$	$\mathbf{.6386 \pm .0062}$	$\mathbf{.7792 \pm .0019}$	$\mathbf{.7149 \pm .0022}$
Random Forest	LDPC	$.3996 \pm .0028$	$.6939 \pm .0049$	$.8338 \pm .0015$	$.7735 \pm .0024$
Gaussian SVM	RREP (BR)	$.3378 \pm .0023$	$.8414 \pm .0051$	$.7955 \pm .0017$	$.7281 \pm .0025$
Gaussian SVM	HAMR	$.2873 \pm .0017$	$.8084 \pm .0047$	$.7681 \pm .0022$	$.7215 \pm .0025$
Gaussian SVM	BCH	$\mathbf{.2550 \pm .0018}$	$\mathbf{.7787 \pm .0048}$	$\mathbf{.7515 \pm .0015}$	$\mathbf{.7053 \pm .0024}$
Gaussian SVM	LDPC	$.3161 \pm .0023$	$.8530 \pm .0041$	$.7963 \pm .0018$	$.7515 \pm .0023$
Logistic Regression	RREP (BR)	$.4821 \pm .0024$	$.7396 \pm .0049$	$.8531 \pm .0016$	$.7458 \pm .0026$
Logistic Regression	HAMR	$.4048 \pm .0020$	$.7170 \pm .0056$	$.8282 \pm .0015$	$.7405 \pm .0024$
Logistic Regression	BCH	$\mathbf{.3291 \pm .0020}$	$\mathbf{.6982 \pm .0048}$	$\mathbf{.8094 \pm .0020}$	$\mathbf{.7205 \pm .0022}$
Logistic Regression	LDPC	$.4659 \pm .0028$	$.7507 \pm .0056$	$.8565 \pm .0019$	$.7694 \pm .0027$
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (BR)	$.0303 \pm .0020$	$.6546 \pm .0025$	$.8872 \pm .0036$	
Random Forest	HAMR	$.0288 \pm .0019$	$.6387 \pm .0025$	$.8851 \pm .0036$	
Random Forest	BCH	$\mathbf{.0250 \pm .0019}$	$\mathbf{.4567 \pm .0034}$	$\mathbf{.8737 \pm .0038}$	
Random Forest	LDPC	$.0312 \pm .0022$	$.5601 \pm .0032$	$.8876 \pm .0035$	
Gaussian SVM	RREP (BR)	$.0273 \pm .0021$	$.3721 \pm .0037$	$.8720 \pm .0041$	
Gaussian SVM	HAMR	$\mathbf{.0243 \pm .0022}$	$.3675 \pm .0036$	$.8718 \pm .0042$	
Gaussian SVM	BCH	$\mathbf{.0255 \pm .0019}$	$\mathbf{.3499 \pm .0030}$	$\mathbf{.8561 \pm .0043}$	
Gaussian SVM	LDPC	$\mathbf{.0243 \pm .0017}$	$.4226 \pm .0034$	$.8782 \pm .0037$	
Logistic Regression	RREP (BR)	$.5084 \pm .0068$	$.5784 \pm .0282$	$.8759 \pm .0035$	
Logistic Regression	HAMR	$.3509 \pm .0089$	$.5499 \pm .0247$	$.8740 \pm .0036$	
Logistic Regression	BCH	$\mathbf{.0295 \pm .0018}$	$\mathbf{.4022 \pm .0076}$	$\mathbf{.8579 \pm .0038}$	
Logistic Regression	LDPC	$.0528 \pm .0031$	$.5396 \pm .0149$	$.8795 \pm .0036$	

them suitably by the ECC is a superior algorithm over the original BR. The micro and macro F_1 scores, and the pairwise label ranking loss are reported in Tables A.12, A.13, and A.14, respectively.

Comparing Tables 2.3 and 2.5, 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 2.4 and 2.6, there is no clear winner between the 3-powerset and BR.

Table 2.6: Hamming loss of ML-ECC using BR base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (BR)	.0858 ± .0005	.1811 ± .0016	.1903 ± .0006	.0662 ± .0003
Random Forest	HAMR	.0728 ± .0005	**.1779 ± .0017**	**.1878 ± .0007**	.0652 ± .0002
Random Forest	BCH	**.0720 ± .0007**	.1828 ± .0018	.1898 ± .0008	**.0638 ± .0003**
Random Forest	LDPC	.0832 ± .0006	.1882 ± .0017	.1963 ± .0006	.0721 ± .0003
Gaussian SVM	RREP (BR)	.0743 ± .0005	**.2460 ± .0021**	**.1866 ± .0006**	.0621 ± .0003
Gaussian SVM	HAMR	**.0717 ± .0004**	**.2480 ± .0023**	**.1861 ± .0007**	**.0616 ± .0003**
Gaussian SVM	BCH	.0738 ± .0005	.2565 ± .0031	.1880 ± .0007	.0619 ± .0003
Gaussian SVM	LDPC	.0742 ± .0006	.2532 ± .0019	.1908 ± .0006	.0688 ± .0003
Logistic Regression	RREP (BR)	.1024 ± .0006	**.2062 ± .0018**	**.2000 ± .0007**	.0641 ± .0003
Logistic Regression	HAMR	**.0959 ± .0006**	**.2049 ± .0022**	**.2003 ± .0007**	**.0635 ± .0003**
Logistic Regression	BCH	**.0955 ± .0006**	.2172 ± .0023	.2037 ± .0008	**.0638 ± .0003**
Logistic Regression	LDPC	.1038 ± .0006	.2133 ± .0019	.2044 ± .0008	.0705 ± .0004
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (BR)	.0013 ± .0001	.0183 ± .0001	**.0474 ± .0003**	
Random Forest	HAMR	.0012 ± .0001	.0179 ± .0001	**.0474 ± .0003**	
Random Forest	BCH	**.0010 ± .0001**	**.0152 ± .0001**	.0494 ± .0004	
Random Forest	LDPC	.0014 ± .0001	.0203 ± .0001	.0529 ± .0004	
Gaussian SVM	RREP (BR)	.0012 ± .0001	.0113 ± .0001	**.0450 ± .0004**	
Gaussian SVM	HAMR	**.0010 ± .0001**	**.0112 ± .0001**	**.0451 ± .0004**	
Gaussian SVM	BCH	**.0010 ± .0001**	.0117 ± .0001	.0487 ± .0005	
Gaussian SVM	LDPC	.0011 ± .0001	.0153 ± .0001	.0517 ± .0005	
Logistic Regression	RREP (BR)	.0347 ± .0007	.0212 ± .0014	**.0455 ± .0003**	
Logistic Regression	HAMR	.0186 ± .0005	.0191 ± .0011	**.0452 ± .0003**	
Logistic Regression	BCH	**.0024 ± .0002**	**.0161 ± .0006**	.0472 ± .0004	
Logistic Regression	LDPC	.0050 ± .0004	.0234 ± .0011	.0517 ± .0004	

Chapter 3

New Decoder for Hard and Soft Decoding of Error-correcting Codes

In Chapter 2 we demonstrated the effectiveness of applying the ECCs on multi-label classification. In addition, we showed that the $\Delta_{0/1}$ is upper bounded by a function of Δ_{HL} in the codewords and the strength of the ECC—the number of bit errors that the ECC is able to correct. However, sometimes in multi-label classification the bit error rate is high. If the codeword prediction of an instance has more bit errors than that the ECC is able to correct, there is no guarantee of the decoding outcome.

The reason is that, the off-the-shelf decoder, e.g., the decoder for the BCH code we used, takes advantages of the algebraic structure of the ECC to locate possible bit errors. The decoder decodes $\tilde{\mathbf{b}} \in \{0,1\}^M$ to $\tilde{\mathbf{y}} \in \{0,1\}^K$ where the encoding of $\tilde{\mathbf{y}}$ is approximately the valid codeword closest to $\tilde{\mathbf{b}}$ in $\{0,1\}^M$. However, when M is large, many possible values of $\tilde{\mathbf{b}}$ are too far away from any valid codeword. If the decoder is able to correct m bit errors, any vertex of the hypercube $\{0,1\}^M$ within m-bit difference from a valid codeword can be perfectly mapped back to a vertex of $\{0,1\}^K$. The number of such vertices is $2^K \cdot \sum_{i=0}^{m} \binom{M}{i}$, which is generally smaller than 2^M. For the other vertices, since they are too far away from any valid codeword, the off-the-shelf decoder cannot utilize the full power of all parity bits but use only k of the M bits, resulting in suboptimal decoding performance. This also explains the sharp increase of $\Delta_{0/1}$ for the BCH code in Figure 2.4(b), in contrast to the smooth slope for LDPC code, which is decoded using Belief Propagation algorithm.

We try to overcome this deficiency by proposing a new decoder in Section 3.1, and experiment on it in Section 3.2. This decoder decodes a vertex of hypercube $\{0,1\}^M$ to the interior of the hypercube $[0,1]^K$, and then round to the nearest vertex of $\{0,1\}^K$. The rounding-based methods have been studied by Tai and Lin [2012]. Because this decoder takes some geometric information into account, we call it *geometric decoder*, and call the off-the-shelf decoder as *algebraic decoder*.

Another benefit of the geometric decoder is to perform interior-to-interior decoding, from $[0,1]^M$ to $[0,1]^K$. In other words, this is a soft-in soft-out decoder [Wolf, 1978, Hagenuaer et al., 1996]. The soft input bits contain the channel measurement information, and the value of each bit represents the confidence in the bit being 1. We discuss about how to gather such information from our channels, the base learners, in Section 3.3 and present experimental results in Section 3.4.

3.1 Geometric Decoder for Linear Codes

Here we describe our proposed geometric decoder in detail. The geometric decoder maps a vertex $\tilde{\mathbf{b}}$ in $\{0,1\}^M$ to a point $\tilde{\mathbf{y}}$ in the interior of $[0,1]^K$. Since the output of this decoder are real values, we call it *soft output*, in contrast to the binary values, which are called *hard output*. As mentioned above, we may convert the soft output of geometric decoder to hard output by rounding.

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 $\mathbf{b} = enc(\mathbf{y}) = G \cdot \mathbf{y}$ (mod 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 $1 \rightarrow -1$ and $0 \rightarrow 1$. 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*.

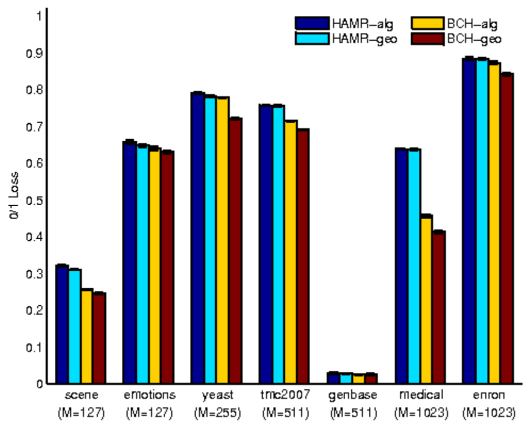
It is difficult to generalize the XOR operation from binary to real values, but multiplication by itself can be defined on real values. We take this advantage and use it to form our geometric decoder. Our geometric decoder would find the $\tilde{\mathbf{y}}$ that minimizes the L_2 distance between $\tilde{\mathbf{b}}$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

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

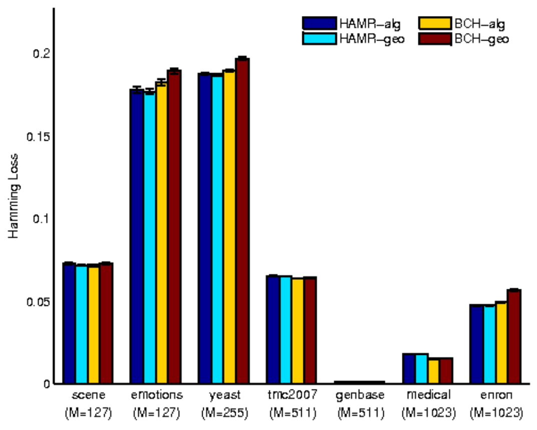
Note that the squared L_2 distance between codewords is an approximation of the Hamming distance in binary space $\{0, 1\}^M$.

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, there is no efficient way 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 has 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, the soft outputs condense the space of valid codewords, so it would be easier to find one close to $\tilde{\mathbf{b}}$.



(a) $0/1$ loss



(b) Hamming loss

Figure 3.1: Hard-input soft-output geometric decoding results for ML-ECC using BR with Random Forests

3.2 Experimental Results of Geometric Decoder

The experiments of the proposed geometric decoder are done on the same setting as that of the off-the-shelf algebraic decoder in Section 2.4. Here, we focus on comparing the new decoder on HAMR and BCH codes with their algebraic decoder since the previous experiments have already shown that these codes are the better choices for multi-label classification.

We first demonstrate the advantage of the proposed geometric decoder over the algebraic one using the same codeword predictions as in Section 2.4. The results are shown in Figure 3.1. Here the base learner is Binary Relevance with Random Forests. In the figures, `alg` stands for the algebraic decoder, and `geo` stands for the proposed geometric decoder. The soft decoding output of the geometric decoder is rounded back to $\{0, 1\}$ for evaluation and comparison.

Figure 3.1(a) shows the result on $0/1$ loss. For the BCH code, the proposed geometric decoder outperforms the algebraic one significantly on almost all datasets, especially the great improvement on the `yeast` and `medical` datasets. For the HAMR code, the geometric decoder is better than the algebraic one except on the `genbase` and `enron` datasets where both decoders have similar $0/1$ loss.

Next we look at the Hamming loss in Figure 3.1(b). For the HAMR code, the proposed

Table 3.1: $0/1$ loss changes when applying the proposed soft-output decoder

ECC	base learner	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
HAMR	BR,Random Forest	$\mathbf{-.0101 \pm .0010}$	$\mathbf{-.0094 \pm .0022}$	$\mathbf{-.0077 \pm .0010}$	$\mathbf{-.0012 \pm .0006}$
HAMR	BR,Gaussian SVM	$\mathbf{-.0047 \pm .0007}$	$\mathbf{-.0145 \pm .0030}$	$\mathbf{-.0031 \pm .0008}$	$\mathbf{-.0009 \pm .0005}$
HAMR	BR,Logistic Regression	$\mathbf{-.0081 \pm .0010}$	$\mathbf{-.0078 \pm .0028}$	$\mathbf{-.0012 \pm .0008}$	$\mathbf{-.0006 \pm .0005}$
HAMR	3-powerset,Random Forest	$\mathbf{-.0099 \pm .0008}$	$\mathbf{-.0071 \pm .0023}$	$\mathbf{-.0101 \pm .0011}$	$\mathbf{-.0014 \pm .0006}$
HAMR	3-powerset,Gaussian SVM	$\mathbf{-.0042 \pm .0006}$	$\mathbf{-.0239 \pm .0029}$	$\mathbf{-.0082 \pm .0006}$	$\mathbf{-.0010 \pm .0007}$
HAMR	3-powerset,Logistic Regression	$\mathbf{-.0064 \pm .0009}$	$\mathbf{-.0041 \pm .0029}$	$\mathbf{-.0051 \pm .0009}$	$-.0004 \pm .0007$
BCH	BR,Random Forest	$\mathbf{-.0100 \pm .0007}$	$\mathbf{-.0101 \pm .0030}$	$\mathbf{-.0575 \pm .0015}$	$\mathbf{-.0231 \pm .0014}$
BCH	BR,Gaussian SVM	$\mathbf{-.0048 \pm .0005}$	$\mathbf{-.0437 \pm .0039}$	$\mathbf{-.0312 \pm .0012}$	$\mathbf{-.0078 \pm .0014}$
BCH	BR,Logistic Regression	$\mathbf{-.0127 \pm .0007}$	$\mathbf{-.0096 \pm .0034}$	$\mathbf{-.0396 \pm .0017}$	$\mathbf{-.0103 \pm .0018}$
BCH	3-powerset,Random Forest	$\mathbf{-.0114 \pm .0007}$	$\mathbf{-.0087 \pm .0032}$	$\mathbf{-.0529 \pm .0016}$	$\mathbf{-.0210 \pm .0013}$
BCH	3-powerset,Gaussian SVM	$\mathbf{-.0048 \pm .0004}$	$\mathbf{-.0343 \pm .0044}$	$\mathbf{-.0250 \pm .0011}$	$\mathbf{-.0090 \pm .0013}$
BCH	3-powerset,Logistic Regression	$\mathbf{-.0070 \pm .0006}$	$\mathbf{-.0114 \pm .0030}$	$\mathbf{-.0256 \pm .0014}$	$\mathbf{-.0122 \pm .0013}$

ECC	base learner	genbase (M=511)	medical (M=1023)	enron (M=1023)
HAMR	BR,Random Forest	$\mathbf{-.0008 \pm .0005}$	$\mathbf{-.0010 \pm .0010}$	$\mathbf{-.0006 \pm .0006}$
HAMR	BR,Gaussian SVM	$\mathbf{-.0012 \pm .0008}$	$-.0004 \pm .0007$	$.0002 \pm .0005$
HAMR	BR,Logistic Regression	$.0032 \pm .0057$	$.0015 \pm .0010$	$.0004 \pm .0004$
HAMR	3-powerset,Random Forest	$.0010 \pm .0005$	$-.0006 \pm .0008$	$-.0001 \pm .0004$
HAMR	3-powerset,Gaussian SVM	$\mathbf{-.0005 \pm .0004}$	$-.0004 \pm .0007$	$.0002 \pm .0005$
HAMR	3-powerset,Logistic Regression	$-.0030 \pm .0061$	$.0004 \pm .0010$	$-.0004 \pm .0005$
BCH	BR,Random Forest	$.0005 \pm .0003$	$\mathbf{-.0438 \pm .0022}$	$\mathbf{-.0308 \pm .0016}$
BCH	BR,Gaussian SVM	$-.0000 \pm .0003$	$\mathbf{-.0068 \pm .0025}$	$\mathbf{-.0133 \pm .0016}$
BCH	BR,Logistic Regression	$.0127 \pm .0018$	$\mathbf{-.0318 \pm .0045}$	$\mathbf{-.0217 \pm .0013}$
BCH	3-powerset,Random Forest	$.0003 \pm .0004$	$\mathbf{-.0280 \pm .0018}$	$\mathbf{-.0238 \pm .0018}$
BCH	3-powerset,Gaussian SVM	$\mathbf{-.0007 \pm .0003}$	$\mathbf{-.0150 \pm .0018}$	$\mathbf{-.0055 \pm .0016}$
BCH	3-powerset,Logistic Regression	$.0003 \pm .0006$	$\mathbf{-.0216 \pm .0022}$	$\mathbf{-.0139 \pm .0013}$

method has a small improvement on the `scene`, `emotions`, and `yeast` datasets, and has similar Hamming loss with the algebraic decoding method on other datasets. However, for the BCH code, the proposed method has worse Hamming loss on the `yeast`, `emotions`, and `enron` datasets. 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 $\{0,1\}^K$).

Similar results show up when using other base learners, as shown in Table 3.1 and 3.2. In the tables, each entry reports the difference between the results of the geometric decoder and the algebraic decoder. The bold entries indicate that the geometric decoder is significantly better than the algebraic one. The results validate that the proposed geomet-

Table 3.2: Hamming loss changes when applying the proposed soft-output decoder

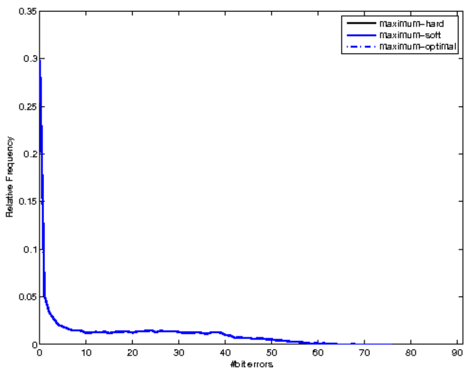
ECC	base learner	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
HAMR	BR,Random Forest	$\mathbf{-.0008 \pm .0002}$	$\mathbf{-.0012 \pm .0007}$	$\mathbf{-.0006 \pm .0002}$	$-.0000 \pm .0001$
HAMR	BR,Gaussian SVM	$-.0001 \pm .0001$	$.0035 \pm .0013$	$\mathbf{-.0003 \pm .0001}$	$\mathbf{-.0001 \pm .0000}$
HAMR	BR,Logistic Regression	$\mathbf{-.0003 \pm .0002}$	$.0018 \pm .0009$	$-.0001 \pm .0002$	$-.0001 \pm .0001$
HAMR	3-powerset,Random Forest	$-.0002 \pm .0002$	$.0005 \pm .0006$	$\mathbf{-.0004 \pm .0002}$	$\mathbf{-.0001 \pm .0000}$
HAMR	3-powerset,Gaussian SVM	$-.0001 \pm .0001$	$.0046 \pm .0009$	$.0001 \pm .0002$	$.0000 \pm .0001$
HAMR	3-powerset,Logistic Regression	$.0001 \pm .0002$	$.0029 \pm .0009$	$.0005 \pm .0002$	$\mathbf{-.0002 \pm .0001}$
BCH	BR,Random Forest	$.0011 \pm .0002$	$.0068 \pm .0009$	$.0070 \pm .0005$	$.0005 \pm .0002$
BCH	BR,Gaussian SVM	$\mathbf{-.0005 \pm .0002}$	$.0192 \pm .0022$	$.0073 \pm .0004$	$.0030 \pm .0001$
BCH	BR,Logistic Regression	$.0002 \pm .0002$	$.0141 \pm .0012$	$.0079 \pm .0006$	$.0036 \pm .0002$
BCH	3-powerset,Random Forest	$.0009 \pm .0002$	$.0035 \pm .0013$	$.0062 \pm .0005$	$.0006 \pm .0002$
BCH	3-powerset,Gaussian SVM	$.0004 \pm .0001$	$.0122 \pm .0023$	$.0055 \pm .0004$	$.0025 \pm .0002$
BCH	3-powerset,Logistic Regression	$.0008 \pm .0002$	$.0089 \pm .0016$	$.0072 \pm .0005$	$.0022 \pm .0002$
ECC	base learner	genbase (M=511)	medical (M=1023)	enron (M=1023)	
HAMR	BR,Random Forest	$\mathbf{-.0000 \pm .0000}$	$\mathbf{-.0000 \pm .0000}$	$-.0000 \pm .0000$	
HAMR	BR,Gaussian SVM	$\mathbf{-.0001 \pm .0000}$	$-.0000 \pm .0000$	$\mathbf{-.0000 \pm .0000}$	
HAMR	BR,Logistic Regression	$.0001 \pm .0004$	$.0001 \pm .0000$	$.0000 \pm .0000$	
HAMR	3-powerset,Random Forest	$.0000 \pm .0000$	$-.0000 \pm .0000$	$\mathbf{-.0001 \pm .0000}$	
HAMR	3-powerset,Gaussian SVM	$\mathbf{-.0000 \pm .0000}$	$-.0000 \pm .0000$	$.0000 \pm .0000$	
HAMR	3-powerset,Logistic Regression	$-.0002 \pm .0002$	$.0000 \pm .0000$	$\mathbf{-.0001 \pm .0000}$	
BCH	BR,Random Forest	$.0000 \pm .0000$	$.0002 \pm .0001$	$.0073 \pm .0003$	
BCH	BR,Gaussian SVM	$.0000 \pm .0000$	$.0009 \pm .0001$	$.0090 \pm .0002$	
BCH	BR,Logistic Regression	$.0030 \pm .0004$	$\mathbf{-.0008 \pm .0003}$	$.0086 \pm .0002$	
BCH	3-powerset,Random Forest	$.0000 \pm .0000$	$.0005 \pm .0001$	$.0083 \pm .0003$	
BCH	3-powerset,Gaussian SVM	$.0000 \pm .0000$	$.0007 \pm .0001$	$.0082 \pm .0004$	
BCH	3-powerset,Logistic Regression	$.0001 \pm .0001$	$-.0001 \pm .0001$	$.0077 \pm .0003$	

ric decoder can decode more accurately (lower $0/1$ loss) and with similar Hamming loss comparing to the algebraic decoder.

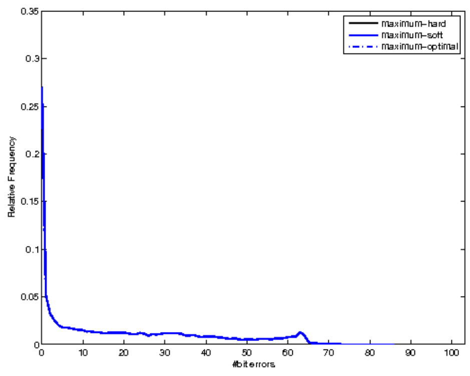
3.2.1 Bit Error Analysis

Next, we look deeper into the `scene` dataset, and fix the base learner to BR with Random Forests. The instances are grouped by the number of bit errors at that instance. First, we plot the ratio of the group size to the total number of instances in Figure 3.2 for HAMR and BCH codes. Besides the highest peak at 0 bit errors, another peak for the BCH code is at 63 bit errors, which is higher than that for HAMR at 38 bit errors. This suggests that BCH code is harder to learn, which is consistent to our finding in Section 2.4.3,

Then, we plot the $0/1$ loss and Hamming loss in each group for HAMR, as shown in Figure 3.3. From Figure 3.3(a), we can see that the geometric decoder is able to correct errors more accurately when there are 16 to 24 bit errors, comparing to the algebraic

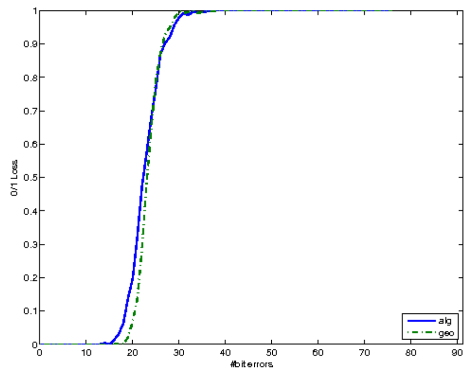


(a) HAMR

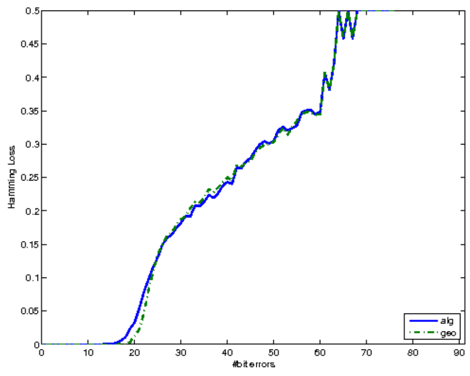


(b) BCH

Figure 3.2: Bit error distribution of BR with Random Forests on the `scene` dataset



(a) $0/1$ loss vs. number of bit errors

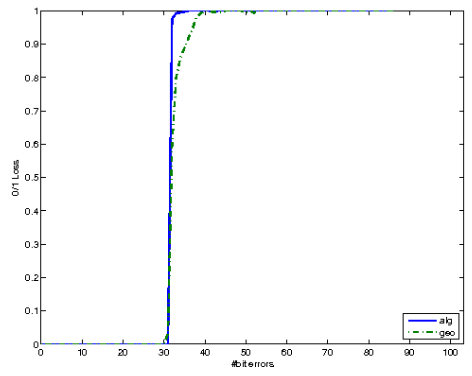


(b) Hamming loss vs. number of bit errors

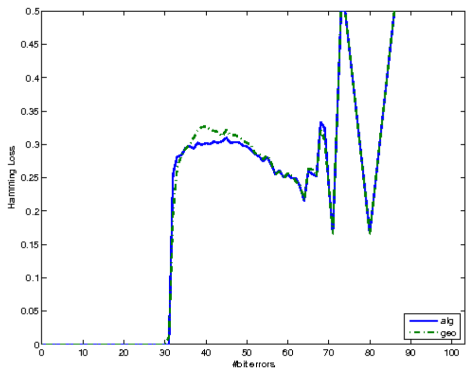
Figure 3.3: Strength of HAMR on the `scene` dataset and BR with Random Forests

decoding. The ordinary decoding method of HAMR has two-stages, one for $HAM(7,4)$ and one for repetition code, and each $HAM(7,4)$ block is decoded independently. In the proposed geometric decoding method, the two stages are combined into one, which enables joint decoding of those $HAM(7,4)$ blocks and thus ensures that the decoding of each $HAM(7,4)$ block is consistent to others. This leads to superior performance of the proposed decoding method on $0/1$ loss. For Hamming loss, as shown in Figure 3.3(b), the improvement of the geometric decoder at that bit error range is small, which explains the small improvement on Hamming loss.

We also plot the $0/1$ loss and Hamming loss in each group for the BCH code in Figure 3.4. The algebraic decoder can correctly recover the label vector with no $0/1$ loss



(a) $0/1$ loss vs. number of bit errors



(b) Hamming loss vs. number of bit errors

Figure 3.4: Strength of BCH on the `scene` dataset and BR with Random Forests

for instances with at most 31 bit errors, but for instance with 32 bit errors the $0/1$ loss sharply goes up to 0.97. The proposed geometric decoder did a better job for instances with 32–39 bit errors, so its $0/1$ loss goes up more smoothly. This is exactly what we would like to address in the beginning of this Chapter. On the other hand, in terms of Hamming loss shown in Figure 3.4(b), the proposed geometric decoder has 0.01–0.025 higher Hamming loss than the algebraic one for instances with 37–45 bit errors, which yields the slightly worse result of geometric decoder on Hamming loss.

From this analysis, we may conclude that the geometric decoder can improve $0/1$ loss because it really does a better job on the instances far from valid codewords. However, regarding Hamming loss, the geometric decoder gets improvements for HAMR, but not for BCH.

3.3 Soft-input Decoding and Bitwise Confidence Estimation for k-powerset Learners

In Section 3.1, we proposed the geometric decoder based on approximating XOR by multiplication. Since L_2 distance in $[0, 1]^M$ space is used as optimization criterion, the input codeword prediction $\tilde{\mathbf{b}}$ is not necessary to be in $\{0, 1\}^M$ but can also be in $[0, 1]^M$. That is, this decoding method supports not only soft outputs but also soft inputs. The soft inputs may come from the confidence of each bit, which the channel, the multi-label base learner, provides. By considering the confidence of bits, the decoder may rely on high-confidence bits more and try to correct low-confidence bits. In this way, the performance of decoders may be further improved.

Since our channel is the base learner, it is possible to gather meaningful soft signals from the channels, which is the confidence score or probability estimate of the predicting bit to be 1. It is simple to ask a Binary Relevance learner to provide confidence of each bit, since confidence or probability estimate is supported by many state-of-the-art binary classifiers, including Random Forests and SVM [Platt, 1999, Lin et al., 2007]. However, for a k-powerset learner, things are more complicated. The k-powerset learners take a combination of k bits as a class, and the base learners only output confidence information per combination of k bits but not per bit. To apply the proposed soft-input geometric decoder, we have to estimate the confidence of each bit from the confidence of the combinations of k bits.

A k-powerset learner would output 2^k confidence scores, one for each combination of the k bits $b_1 \cdots b_k \in \{0,1\}^k$. To estimate per-bit confidence $conf(b_i = 1)$ for each b_i, we propose the following methods.

1. **Maximum**. Pick the combination $b_1^* \cdots b_k^*$ with the highest confidence and then assign $conf(b_i = 1)$ to be 1 if $b_i^* = 1$, or 0 otherwise. This results in the hard input, which is the same as what we used in Section 2.4 and 3.2.

2. **Marginal probability**. The confidence score of each combination can be treated as the joint probability distributed over the 2^K combinations of bits. Then, we may calculate $conf(b_i = 1)$ as marginal probability by summing up the confidence scores of all combinations with $b_i = 1$.

$$conf_{margin}(b_i) = \sum_{b_1 \cdots b_{i-1} b_{i+1} \cdots b_k \in \{0,1\}^{k-1}} conf(b_1 \cdots b_{i-1} 1 b_{i+1} \cdots b_k)$$

In contrast to the "maximum" method, the marginal probability takes the whole distribution into account, so the most probable combination according to marginal probability may be different from the one with the highest confidence.

3. **Confidence difference**. The confidence of the ith bit to be b_i^* may be defined as the difference of confidence scores between the most confident combination $b_1^* \cdots b_k^*$ and its neighbor varying only this bit $b_1^* \cdots b_{i-1}^* \overline{b_i^*} b_{i+1}^* \cdots b_k^*$, where $\overline{b_i^*}$ is the negation of b_i^*. Following the idea, we may define $conf(b_i = 1)$ as

$$conf_{diff}(b_i) = \frac{1}{2} + \frac{1}{2}\left(conf(b_1^* \cdots b_{i-1}^* 1 b_{i+1}^* \cdots b_k^*) - conf(b_1^* \cdots b_{i-1}^* 0 b_{i+1}^* \cdots b_k^*)\right)$$

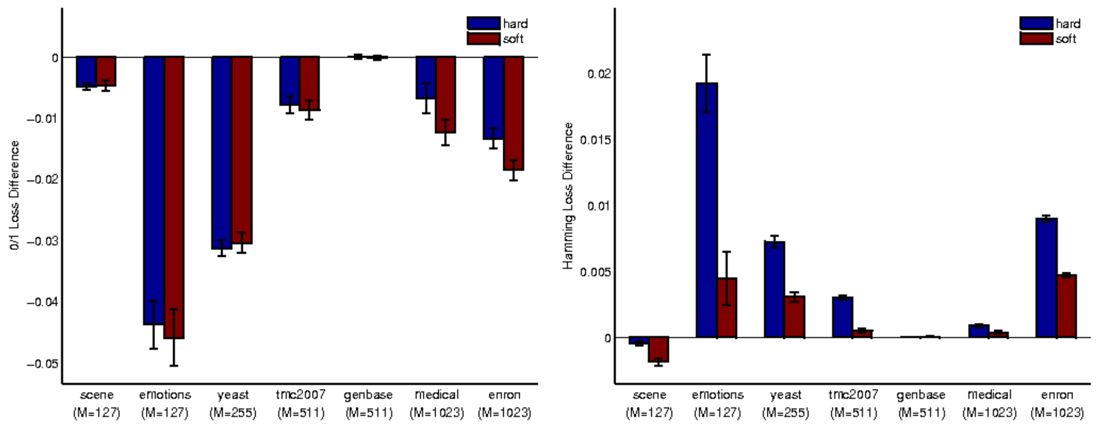
   Comparing to "marginal probability," which is essentially the sum of difference of confidence scores between all pairs of neighboring combinations, this method only considers the highest-confidence combination and its neighbors. Therefore, the result of "confidence difference" is consistent with the "maximum" method.

4. **Sigmoid functions**. We may apply sigmoid functions on the "confidence difference" to enlarge the small amount of difference. The reason is that small confidence values make the geometric decoder not stable. We used $\tanh(\alpha x)$ as the sigmoid function.

$$conf_{s\text{-}diff}(b_i) = \frac{1}{2} + \frac{1}{2}\tanh\left(\alpha \cdot (2 \cdot conf_{diff}(b_i) - 1)\right)$$

   The sigmoid functions may also be applied on the output of "marginal probability," resulting in another confidence estimating method $conf_{s\text{-}margin}(\cdot)$.

Note that the Binary Relevance approach is a special case of k-powerset with $k = 1$. Therefore, we may also apply these methods for BR learners. When applying to BR learners, the "marginal probability" would be the same as taking the confidence of the bit directly from the BR learner. Moreover, if the confidence is given in probability form, the "confidence difference" would also be the same.



(a) $0/1$ loss change (b) Hamming loss change

Figure 3.5: Comparison between hard-/soft-input geometric decoders in the ML-ECC with the BCH code using BR learners

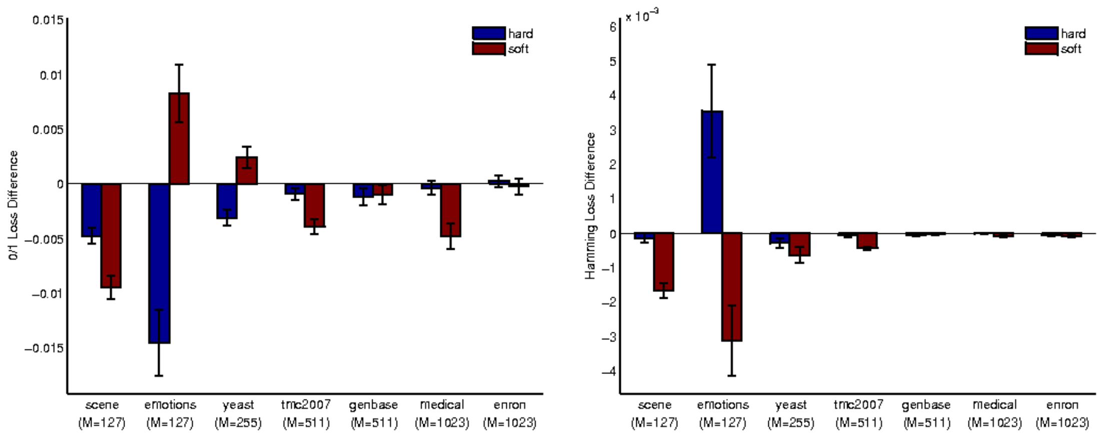
3.4 Experimental Results of Soft-input Geometric Decoder

Now, we experimentally compare the soft-input geometric decoder with the hard-input one. The settings of these experiments are the same as in Section 2.4 and 3.2.

3.4.1 Soft-input Decoding for Binary Relevance Learners

All of the base learners we used, Random Forests from WEKA, Gaussian SVM from LIBSVM, and logistic regression from LIBLINEAR, support predicting class probability distribution. To take the class probability distribution as soft inputs, we first try on Binary Relevance approaches. In the Binary Relevance approach, each base learner learns a single bit, so its probability output is indeed the soft signal we want.

The results on the BCH code using the Gaussian SVM base learner is shown in Figure 3.5. Since the value of $\Delta_{0/1}$ and Δ_{HL} varies greatly from dataset to dataset but little from decoder to decoder, in the figures, we present the $\Delta_{0/1}$ and Δ_{HL} changes based on the results of the algebraic decoder. We denote the result of hard-input geometric decoder as `hard`, and that of soft-input geometric decoder as `soft`. The value lower than 0 means that the geometric decoder performs better than the algebraic one. We can see from Figure 3.5(a) that the soft-input geometric decoder is similar to or slightly better than the hard-input one in terms of $\Delta_{0/1}$, and both geometric decoders are significantly



(a) $0/1$ loss change (b) Hamming loss change

Figure 3.6: Comparison between hard-/soft-input geometric decoders in the ML-ECC with the HAMR code using BR learners

better than the algebraic one. From Figure 3.5(b), we can see that soft-input geometric decoder is much better than the hard-input one in terms of Δ_{HL}, and it is even better than the algebraic one on the `scene` dataset. The result suggests that soft inputs are helpful on Δ_{HL} and also $\Delta_{0/1}$ for geometric decoder and for BCH code.

In contrast to the BCH code, the results on the HAMR code is a little different, as shown in Figure 3.6. First, we look at the $0/1$ loss shown in Figure 3.6(a). The result is dataset dependent. Soft-input geometric decoder performs better than both hard-input one and algebraic one on three datasets, and worse than those two decoders on other two datasets. In terms of Hamming loss in Figure 3.6(b), soft-input geometric decoder is significantly better than both hard-input one and algebraic one on four datasets, and performs similar on the other datasets. The result indicates that soft inputs are helpful on Δ_{HL}, but not on $\Delta_{0/1}$, when applying to HAMR code.

Similar results show up when using other base learners, as shown in Table 3.3 and 3.4. The bold-face entries are the best entries on each dataset given the ECC and base learner. We also report the micro and macro F_1 scores, and the pairwise label ranking loss in Tables A.15, A.16, and A.17, respectively. In the tables, we can see that the soft-input geometric decoder is usually better than the hard-input ones in these measures. This again shows that the soft inputs, i.e. confidence information from base learners, are useful on decoding.

Table 3.3: $0/1$ 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)
Random Forest	HAMR	alg-hard	$.3212 \pm .0021$	$.6574 \pm .0050$	$.7910 \pm .0020$	$\mathbf{.7578 \pm .0025}$
Random Forest	HAMR	geo-hard	$\mathbf{.3111 \pm .0021}$	$\mathbf{.6480 \pm .0051}$	$\mathbf{.7833 \pm .0022}$	$\mathbf{.7566 \pm .0026}$
Random Forest	HAMR	geo-soft	$.3294 \pm .0021$	$.6584 \pm .0053$	$.7976 \pm .0020$	$\mathbf{.7588 \pm .0025}$
Gaussian SVM	HAMR	alg-hard	$.2876 \pm .0018$	$.8073 \pm .0043$	$.7681 \pm .0022$	$.7215 \pm .0025$
Gaussian SVM	HAMR	geo-hard	$.2829 \pm .0017$	$\mathbf{.7927 \pm .0051}$	$\mathbf{.7650 \pm .0023}$	$.7205 \pm .0024$
Gaussian SVM	HAMR	geo-soft	$\mathbf{.2782 \pm .0016}$	$.8155 \pm .0048$	$.7705 \pm .0021$	$\mathbf{.7176 \pm .0022}$
Logistic Regression	HAMR	alg-hard	$.4050 \pm .0020$	$.7175 \pm .0054$	$\mathbf{.8282 \pm .0015}$	$.7405 \pm .0024$
Logistic Regression	HAMR	geo-hard	$.3969 \pm .0022$	$\mathbf{.7097 \pm .0059}$	$\mathbf{.8270 \pm .0016}$	$\mathbf{.7399 \pm .0024}$
Logistic Regression	HAMR	geo-soft	$\mathbf{.3875 \pm .0024}$	$.7177 \pm .0069$	$\mathbf{.8284 \pm .0016}$	$\mathbf{.7379 \pm .0024}$
Random Forest	BCH	alg-hard	$.2562 \pm .0020$	$.6404 \pm .0060$	$.7792 \pm .0019$	$.7149 \pm .0022$
Random Forest	BCH	geo-hard	$\mathbf{.2462 \pm .0019}$	$\mathbf{.6304 \pm .0049}$	$\mathbf{.7217 \pm .0022}$	$\mathbf{.6917 \pm .0020}$
Random Forest	BCH	geo-soft	$.2526 \pm .0020$	$\mathbf{.6264 \pm .0049}$	$.7287 \pm .0022$	$.6949 \pm .0024$
Gaussian SVM	BCH	alg-hard	$.2552 \pm .0018$	$.7809 \pm .0050$	$.7515 \pm .0015$	$.7053 \pm .0024$
Gaussian SVM	BCH	geo-hard	$\mathbf{.2503 \pm .0018}$	$\mathbf{.7371 \pm .0051}$	$\mathbf{.7203 \pm .0018}$	$\mathbf{.6975 \pm .0025}$
Gaussian SVM	BCH	geo-soft	$\mathbf{.2505 \pm .0019}$	$\mathbf{.7350 \pm .0050}$	$\mathbf{.7212 \pm .0021}$	$\mathbf{.6966 \pm .0030}$
Logistic Regression	BCH	alg-hard	$.3291 \pm .0020$	$.6982 \pm .0048$	$.8094 \pm .0020$	$.7205 \pm .0022$
Logistic Regression	BCH	geo-hard	$.3164 \pm .0017$	$.6886 \pm .0046$	$\mathbf{.7698 \pm .0019}$	$\mathbf{.7102 \pm .0021}$
Logistic Regression	BCH	geo-soft	$\mathbf{.3142 \pm .0016}$	$\mathbf{.6787 \pm .0046}$	$\mathbf{.7713 \pm .0023}$	$\mathbf{.7112 \pm .0025}$

base learner	ECC	decoder	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	HAMR	alg-hard	$\mathbf{.0288 \pm .0019}$	$\mathbf{.6387 \pm .0025}$	$\mathbf{.8851 \pm .0036}$
Random Forest	HAMR	geo-hard	$\mathbf{.0280 \pm .0019}$	$\mathbf{.6377 \pm .0027}$	$\mathbf{.8845 \pm .0036}$
Random Forest	HAMR	geo-soft	$\mathbf{.0271 \pm .0017}$	$\mathbf{.6373 \pm .0027}$	$\mathbf{.8848 \pm .0036}$
Gaussian SVM	HAMR	alg-hard	$\mathbf{.0243 \pm .0022}$	$.3675 \pm .0036$	$\mathbf{.8718 \pm .0042}$
Gaussian SVM	HAMR	geo-hard	$\mathbf{.0231 \pm .0021}$	$.3671 \pm .0036$	$\mathbf{.8720 \pm .0042}$
Gaussian SVM	HAMR	geo-soft	$\mathbf{.0233 \pm .0022}$	$\mathbf{.3627 \pm .0035}$	$\mathbf{.8716 \pm .0043}$
Logistic Regression	HAMR	alg-hard	$.3509 \pm .0089$	$\mathbf{.5499 \pm .0247}$	$\mathbf{.8740 \pm .0036}$
Logistic Regression	HAMR	geo-hard	$.3541 \pm .0099$	$\mathbf{.5514 \pm .0249}$	$\mathbf{.8744 \pm .0036}$
Logistic Regression	HAMR	geo-soft	$\mathbf{.2777 \pm .0107}$	$\mathbf{.5301 \pm .0229}$	$\mathbf{.8723 \pm .0036}$
Random Forest	BCH	alg-hard	$\mathbf{.0250 \pm .0019}$	$.4567 \pm .0034$	$.8737 \pm .0038$
Random Forest	BCH	geo-hard	$\mathbf{.0255 \pm .0018}$	$\mathbf{.4130 \pm .0037}$	$.8429 \pm .0038$
Random Forest	BCH	geo-soft	$\mathbf{.0250 \pm .0018}$	$\mathbf{.4157 \pm .0037}$	$\mathbf{.8371 \pm .0043}$
Gaussian SVM	BCH	alg-hard	$\mathbf{.0255 \pm .0019}$	$.3499 \pm .0030$	$.8561 \pm .0043$
Gaussian SVM	BCH	geo-hard	$\mathbf{.0255 \pm .0019}$	$.3431 \pm .0034$	$.8428 \pm .0045$
Gaussian SVM	BCH	geo-soft	$\mathbf{.0253 \pm .0020}$	$\mathbf{.3376 \pm .0035}$	$\mathbf{.8376 \pm .0045}$
Logistic Regression	BCH	alg-hard	$\mathbf{.0295 \pm .0018}$	$.4022 \pm .0076$	$.8579 \pm .0038$
Logistic Regression	BCH	geo-hard	$.0422 \pm .0026$	$\mathbf{.3704 \pm .0048}$	$\mathbf{.8362 \pm .0040}$
Logistic Regression	BCH	geo-soft	$.0395 \pm .0026$	$\mathbf{.3670 \pm .0046}$	$.8475 \pm .0040$

3.4.2 Soft-input Decoding for k-powerset Learners

We have shown that soft inputs are beneficial for geometric decoder when using Binary Relevance base learners. Now, we would like to see if we can apply this to the k-powerset learners. As mentioned in Section 3.3, it is non-trivial to estimate the confidence per bit from the confidence information on k-powerests, i.e., the probability distribution over 2^K

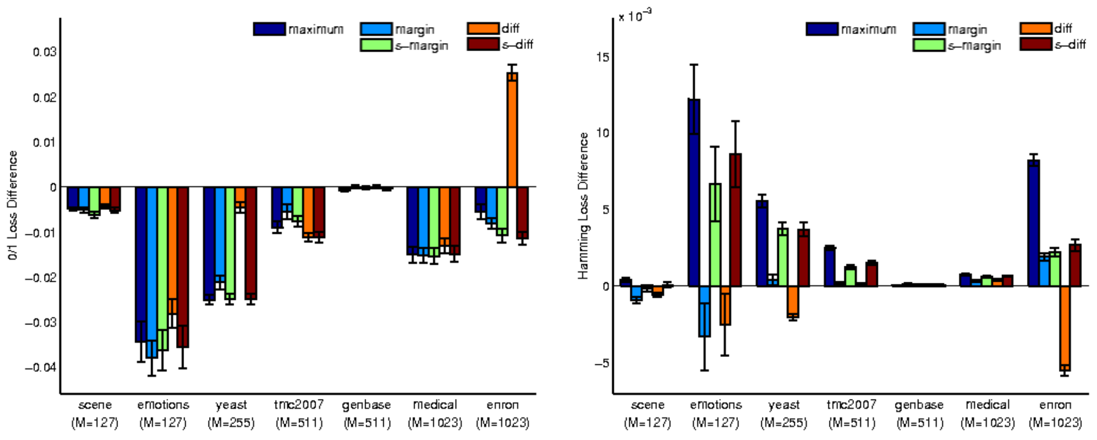
Table 3.4: Hamming loss of ML-ECC with hard-/soft-input geometric decoders and BR

base learner	ECC	decoder	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	HAMR	alg-hard	.0726 ± .0005	**.1781 ± .0017**	**.1878 ± .0007**	**.0652 ± .0002**
Random Forest	HAMR	geo-hard	**.0718 ± .0006**	**.1768 ± .0017**	**.1872 ± .0007**	**.0651 ± .0003**
Random Forest	HAMR	geo-soft	.0726 ± .0005	**.1763 ± .0018**	**.1873 ± .0007**	**.0651 ± .0002**
Gaussian SVM	HAMR	alg-hard	.0717 ± .0004	.2472 ± .0023	**.1861 ± .0007**	.0616 ± .0003
Gaussian SVM	HAMR	geo-hard	.0716 ± .0005	.2508 ± .0023	**.1858 ± .0007**	.0615 ± .0003
Gaussian SVM	HAMR	geo-soft	**.0701 ± .0004**	**.2441 ± .0020**	**.1855 ± .0007**	**.0611 ± .0003**
Logistic Regression	HAMR	alg-hard	.0959 ± .0006	**.2047 ± .0022**	.2003 ± .0007	.0635 ± .0003
Logistic Regression	HAMR	geo-hard	.0956 ± .0006	.2065 ± .0023	.2002 ± .0007	.0634 ± .0003
Logistic Regression	HAMR	geo-soft	**.0920 ± .0006**	**.2032 ± .0024**	**.1990 ± .0007**	**.0631 ± .0003**
Random Forest	BCH	alg-hard	.0717 ± .0006	**.1826 ± .0018**	**.1898 ± .0008**	.0638 ± .0003
Random Forest	BCH	geo-hard	.0728 ± .0006	.1895 ± .0017	.1968 ± .0010	.0643 ± .0003
Random Forest	BCH	geo-soft	**.0703 ± .0006**	**.1822 ± .0016**	.1910 ± .0008	**.0622 ± .0003**
Gaussian SVM	BCH	alg-hard	.0739 ± .0006	**.2569 ± .0030**	**.1880 ± .0007**	**.0619 ± .0003**
Gaussian SVM	BCH	geo-hard	.0735 ± .0005	.2761 ± .0031	.1952 ± .0007	.0649 ± .0003
Gaussian SVM	BCH	geo-soft	**.0721 ± .0006**	.2614 ± .0028	.1911 ± .0007	.0624 ± .0003
Logistic Regression	BCH	alg-hard	.0955 ± .0006	**.2172 ± .0023**	**.2037 ± .0008**	**.0638 ± .0003**
Logistic Regression	BCH	geo-hard	.0957 ± .0006	.2312 ± .0027	.2116 ± .0007	.0673 ± .0003
Logistic Regression	BCH	geo-soft	**.0913 ± .0006**	**.2170 ± .0026**	.2067 ± .0008	**.0640 ± .0003**

base learner	ECC	decoder	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	HAMR	alg-hard	**.0012 ± .0001**	**.0179 ± .0001**	**.0474 ± .0003**
Random Forest	HAMR	geo-hard	**.0012 ± .0001**	**.0179 ± .0001**	**.0474 ± .0003**
Random Forest	HAMR	geo-soft	**.0011 ± .0001**	**.0179 ± .0001**	**.0474 ± .0003**
Gaussian SVM	HAMR	alg-hard	**.0010 ± .0001**	**.0112 ± .0001**	**.0451 ± .0004**
Gaussian SVM	HAMR	geo-hard	**.0009 ± .0001**	**.0112 ± .0001**	**.0450 ± .0004**
Gaussian SVM	HAMR	geo-soft	**.0010 ± .0001**	**.0111 ± .0001**	**.0450 ± .0004**
Logistic Regression	HAMR	alg-hard	.0186 ± .0005	.0191 ± .0011	**.0452 ± .0003**
Logistic Regression	HAMR	geo-hard	.0187 ± .0006	.0192 ± .0011	**.0453 ± .0003**
Logistic Regression	HAMR	geo-soft	**.0134 ± .0005**	**.0180 ± .0010**	**.0453 ± .0003**
Random Forest	BCH	alg-hard	**.0010 ± .0001**	.0152 ± .0001	**.0494 ± .0004**
Random Forest	BCH	geo-hard	**.0010 ± .0001**	.0154 ± .0001	.0566 ± .0005
Random Forest	BCH	geo-soft	**.0010 ± .0001**	**.0150 ± .0002**	.0535 ± .0004
Gaussian SVM	BCH	alg-hard	**.0010 ± .0001**	**.0117 ± .0001**	**.0487 ± .0005**
Gaussian SVM	BCH	geo-hard	**.0010 ± .0001**	.0126 ± .0001	.0577 ± .0006
Gaussian SVM	BCH	geo-soft	**.0011 ± .0001**	.0121 ± .0001	.0534 ± .0005
Logistic Regression	BCH	alg-hard	**.0024 ± .0002**	.0161 ± .0006	**.0472 ± .0004**
Logistic Regression	BCH	geo-hard	.0054 ± .0004	.0154 ± .0004	.0558 ± .0004
Logistic Regression	BCH	geo-soft	.0046 ± .0003	**.0141 ± .0003**	.0526 ± .0004

combinations of labels. Here, we experimentally examine the methods for such estimation described in Section 3.3.

The results on the BCH code are shown in Figure 3.7. Similar to the above experiments, we also plot the changes on $\Delta_{0/1}$ and Δ_{HL} based on the result of the algebraic decoder. In the figures, `maximum`, `margin`, `diff`, `s-margin`, and `s-diff` stand for the soft-input geometric decoders using the corresponding confidence estimation methods described in Section 3.3. Note that `maximum` is exactly the same as supplying hard-input



(a) $0/1$ loss change (b) Hamming loss change

Figure 3.7: Comparison between hard-/soft-input geometric decoders in the ML-ECC with the BCH code using 3-powerset learners

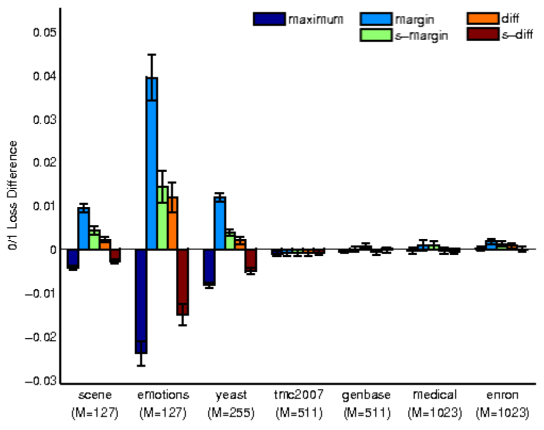
to the geometric decoder.

From Figure 3.7(a), we can see that most of the soft-input geometric decoders have similar performance to the hard-input geometric decoder on $\Delta_{0/1}$, except `diff`, whose result is worse than the hard-input one. The result implies that such bitwise soft inputs calculated from confidence score on k-powersets may not be useful to improve geometric decoding result in terms of $\Delta_{0/1}$.

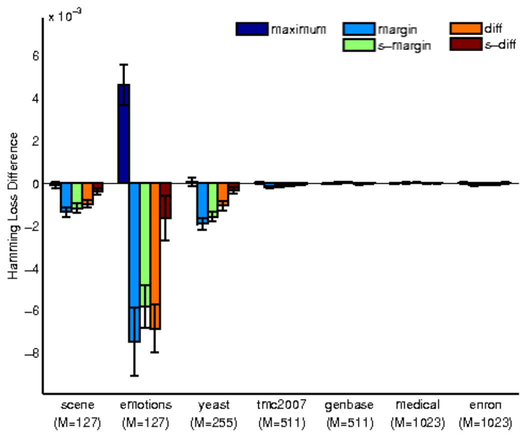
Next, we look at the Hamming loss shown in Figure 3.7(b). On the contrary, soft-input geometric decoders using any confidence estimation method are better than hard-input geometric decoder on Δ_{HL}. Among the confidence estimation methods, `diff` is the best and `margin` is the second best. These two soft-input geometric decoders are even better than the algebraic decoder in terms of Δ_{HL} on some datasets.

Putting the results on $\Delta_{0/1}$ and Δ_{HL} together, we can see that for the BCH code, the soft-input geometric decoder using the "marginal probability" method is a good choice since it has better Δ_{HL} and similar $\Delta_{0/1}$ comparing to the hard-input geometric decoder.

However, things are different for the HAMR code, as shown in Figure 3.8. From Figure 3.8(a), the hard-input geometric decoder beats all soft-input ones in terms of $\Delta_{0/1}$. In the soft-input geometric decoders, `s-diff` has the best performance. This one is also the only soft-input geometric decoder, which is better than the algebraic decoder.



(a) $0/1$ loss change



(b) Hamming loss change

Figure 3.8: Comparison between hard-/soft-input geometric decoders in the ML-ECC with the HAMR code using 3-powerset learners

Table 3.5: Comparison of soft-input geometric decoders using different confidence estimation methods on k-powerset learners

code	measure	order ($\succ$: better than; $\approx$: similar to)
BCH	$\Delta_{0/1}$	*hard* $\approx$ `margin` $\approx$ `s-margin` $\approx$ `s-diff` $\succ$ *algebraic* $\succ$ `diff`
BCH	Δ_{HL}	`diff` $\succ$ `margin` $\succ$ *algebraic* $\succ$ `s-margin` $\approx$ `s-diff` $\succ$ *hard*
HAMR	$\Delta_{0/1}$	*hard* $\succ$ `s-diff` $\succ$ *algebraic* $\succ$ `diff` $\succ$ `s-margin` $\succ$ `margin`
HAMR	Δ_{HL}	`margin` $\approx$ `s-margin` $\approx$ `diff` $\succ$ `s-diff` $\succ$ *algebraic* $\succ$ *hard*

The result is turned over when looking at Δ_{HL} in Figure 3.8(b). All soft-input geometric decoders perform better than the hard-input geometric decoder and the algebraic decoder, but `s-diff` is the worst among the soft-input geometric decoders.

The results on other datasets are similar, and we summarize them in Table 3.5. The results have demonstrated that the performance of the soft-input methods is dependent to the code. In addition, the two evaluation measures, $\Delta_{0/1}$ and Δ_{HL}, focus on different aspects of multi-label classification and different soft-input methods take different trade-off between these two measures. It is still an open problem to design a good bitwise confidence estimation method suitable for both $\Delta_{0/1}$ and Δ_{HL}, or to decode directly from the confidence on k-powersets.

3.5 Comparison with Real-valued ECC

Our ML-ECC framework only considers binary ECC. In this section, we compare our ML-ECC framework with real-valued ECC methods: coding with canonical correlation analysis (CCA-OC) [Zhang and Schneider, 2011] and max-margin output coding (Max-Margin) [Zhang and Schneider, 2012]. The former method uses canonical correlation analysis to find the most linearly-predictable transform of original labels. The latter one uses metric learning to locate a good encoding function. Both methods uses approximate inference as their decoding method.

The main difference between the real-valued ECC and our ML-ECC framework is that our encoding functions transform the label vector into a binary codeword, while the real-valued ECC methods transform the label vector into a real-valued codeword. Moreover, the base learners in our framework deal with classification tasks, while the base learners in those real-valued ECC methods deal with regression tasks.

The experiment setting is basically the same as in Section 2.4, 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.

The results on Random Forests learners are shown in Table 3.6, and the results on logistic regression learners are shown in Table 3.7. 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

Table 3.6: 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	.3294 ± .0021	.0726 ± .0005	.7710 ± .0018	.7741 ± .0017
BCH-geo-soft	**.2526 ± .0020**	**.0703 ± .0006**	**.7944 ± .0019**	**.8006 ± .0018**
CCA-OC	.3165 ± .0010	.0934 ± .0003	.7319 ± .0010	.7403 ± .0010
		emotions		
ECC	0/1 loss	Hamming loss	Micro-F_1	Macro-F_1
HAMR-geo-soft	.6584 ± .0053	**.1763 ± .0018**	.7023 ± .0031	.6756 ± .0034
BCH-geo-soft	**.6264 ± .0049**	.1822 ± .0016	**.7153 ± .0025**	**.6975 ± .0025**
CCA-OC	.6728 ± .0021	.2022 ± .0009	.6920 ± .0014	.6824 ± .0014

Table 3.7: 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	.3875 ± .0024	.0920 ± .0006	.7156 ± .0019	.7204 ± .0017
BCH-geo-soft	**.3142 ± .0016**	**.0913 ± .0006**	**.7337 ± .0016**	**.7397 ± .0014**
CCA-OC	.3599 ± .0011	.1088 ± .0004	.6875 ± .0011	.6952 ± .0011
MaxMargin	.3654 ± .0023	.1107 ± .0009	.6820 ± .0024	.6889 ± .0025
		emotions		
ECC	0/1 loss	Hamming loss	Micro-F_1	Macro-F_1
HAMR-geo-soft	.7177 ± .0069	**.2032 ± .0024**	.6544 ± .0045	.6310 ± .0047
BCH-geo-soft	**.6787 ± .0046**	.2170 ± .0026	.6652 ± .0043	.6499 ± .0044
CCA-OC	.6814 ± .0024	.2068 ± .0006	**.6791 ± .0008**	**.6715 ± .0010**
MaxMargin	.6855 ± .0030	.2099 ± .0009	.6768 ± .0013	.6679 ± .0014

that the power of logistic regression base learner is limited, and the parity bits of HAMR and BCH are too sophisticated for the base learner. On the other hand, the code generated by CCA-OC and MaxMargin methods are easier to learn for such linear model. For sufficiently sophisticated base learners, the proposed discrete-ECC-based framework is the better choice for multi-label classification with error correcting codes.

Chapter 4

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.

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 [Schapire, 1997, Li, 2006].

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. We also proposed and studied several methods to gather soft input from the k-powerset learner. The results show that different ECC designs match different methods better. It remains an interesting research problem on appropriately calculating the 1-bit confidence from the k-bit confidence.

Bibliography

R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1), 1960.

M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9), 2004.

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`.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence in multi-label classification. In *Proc. of the 2nd International Workshop on Learning from Multi-Label Data*, 2010.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2, 1995.

S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas. Protein classification with multiple algorithms. In *Proc. of Panhellenic Conference on Informatics*, 2005.

A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In *Advances in Neural Information Processing Systems 14*, 2002.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008.

R. G. Gallager. *Low Density Parity Check Codes, Monograph.* M.I.T. Press, 1963.

J. Hagenuaer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42(2), 1996.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1), 2009.

R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2), 1950.

A. Hocquenghem. Codes correcteurs d'erreurs. *Chiffres*, 2, 1959.

D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems 22*, 2009.

A. Z. Kouzani. Multilabel classification using error correction codes. In *Advances in Computation and Intelligence - 5th International Symposium*, 2010.

A. Z. Kouzani and G. Nasireding. Multilabel classification by BCH code and random forests. *International Journal of Recent Trends in Engineering*, 2(1), 2009.

L. Li. Multiclass boosting with repartitioning. In *Proc. of the 23rd International Conference on Machine Learning*, 2006.

H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3), 2007.

D. J. C. Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1st edition, 2003.

J. P. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. A shared task involving multi-label classification of clinical free text. In *Proc. of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing*, 2007.

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.

R. E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. of the 14th International Conference on Machine Learning*, 1997.

C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27, 1948.

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

K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proc. of the 9th International Conference on Music Information Retrieval*, 2008.

G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Proc. of the 18th European Conference on Machine Learning*, 2007.

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

J. K. Wolf. Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Transactions on Information Theory*, 24(1), 1978.

Y. Zhang and J. Schneider. Multi-label output codes using canonical correlation analysis. In *Proc. of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.

Y. Zhang and J. Schneider. Maximum margin output coding. In *Proc. of the 29th International Conference on Machine Learning*, 2012.

Appendix A

Additional Experiment Results

A.1 ML-ECC Using 3-powerset Base Learners

Table A.1: Micro-F_1 of ML-ECC using 3-powerset base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (RAKEL)	$.7620 \pm .0019$	$\mathbf{.7058} \pm \mathbf{.0031}$	$.6548 \pm .0016$	$.5988 \pm .0017$
Random Forest	HAMR	$\mathbf{.7777} \pm \mathbf{.0019}$	$\mathbf{.7078} \pm \mathbf{.0032}$	$\mathbf{.6615} \pm \mathbf{.0015}$	$.6025 \pm .0016$
Random Forest	BCH	$\mathbf{.7793} \pm \mathbf{.0019}$	$.7029 \pm .0033$	$.6537 \pm .0016$	$\mathbf{.6172} \pm \mathbf{.0019}$
Random Forest	LDPC	$.7586 \pm .0022$	$.6816 \pm .0037$	$.6300 \pm .0016$	$.5787 \pm .0021$
Gaussian SVM	RREP (RAKEL)	$.7883 \pm .0014$	$\mathbf{.5716} \pm \mathbf{.0039}$	$\mathbf{.6743} \pm \mathbf{.0013}$	$\mathbf{.6618} \pm \mathbf{.0018}$
Gaussian SVM	HAMR	$\mathbf{.7907} \pm \mathbf{.0013}$	$\mathbf{.5747} \pm \mathbf{.0040}$	$\mathbf{.6753} \pm \mathbf{.0012}$	$\mathbf{.6619} \pm \mathbf{.0018}$
Gaussian SVM	BCH	$.7865 \pm .0016$	$.5602 \pm .0042$	$.6687 \pm .0013$	$.6495 \pm .0019$
Gaussian SVM	LDPC	$.7817 \pm .0016$	$.5222 \pm .0057$	$.6611 \pm .0015$	$.6362 \pm .0018$
Logistic Regression	RREP (RAKEL)	$.7232 \pm .0016$	$\mathbf{.6630} \pm \mathbf{.0044}$	$.6445 \pm .0014$	$\mathbf{.6458} \pm \mathbf{.0022}$
Logistic Regression	HAMR	$\mathbf{.7321} \pm \mathbf{.0015}$	$\mathbf{.6597} \pm \mathbf{.0043}$	$\mathbf{.6476} \pm \mathbf{.0013}$	$.6433 \pm .0019$
Logistic Regression	BCH	$\mathbf{.7319} \pm \mathbf{.0016}$	$.6408 \pm .0041$	$.6392 \pm .0015$	$.6309 \pm .0020$
Logistic Regression	LDPC	$.7095 \pm .0020$	$.6292 \pm .0034$	$.6340 \pm .0019$	$.6231 \pm .0024$
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (RAKEL)	$.9870 \pm .0009$	$.5474 \pm .0023$	$\mathbf{.5400} \pm \mathbf{.0050}$	
Random Forest	HAMR	$.9872 \pm .0011$	$.5567 \pm .0025$	$\mathbf{.5376} \pm \mathbf{.0051}$	
Random Forest	BCH	$\mathbf{.9885} \pm \mathbf{.0010}$	$\mathbf{.6786} \pm \mathbf{.0030}$	$.5313 \pm .0048$	
Random Forest	LDPC	$.9858 \pm .0011$	$.6125 \pm .0033$	$.5126 \pm .0057$	
Gaussian SVM	RREP (RAKEL)	$.9864 \pm .0011$	$\mathbf{.7837} \pm \mathbf{.0020}$	$\mathbf{.5690} \pm \mathbf{.0053}$	
Gaussian SVM	HAMR	$.9862 \pm .0013$	$\mathbf{.7837} \pm \mathbf{.0019}$	$\mathbf{.5679} \pm \mathbf{.0055}$	
Gaussian SVM	BCH	$\mathbf{.9888} \pm \mathbf{.0009}$	$.7777 \pm .0021$	$.5217 \pm .0064$	
Gaussian SVM	LDPC	$.9850 \pm .0014$	$.7414 \pm .0020$	$.5251 \pm .0059$	
Logistic Regression	RREP (RAKEL)	$.8383 \pm .0041$	$.7130 \pm .0093$	$\mathbf{.5668} \pm \mathbf{.0035}$	
Logistic Regression	HAMR	$.9012 \pm .0039$	$.7259 \pm .0078$	$\mathbf{.5639} \pm \mathbf{.0036}$	
Logistic Regression	BCH	$\mathbf{.9863} \pm \mathbf{.0011}$	$\mathbf{.7514} \pm \mathbf{.0029}$	$.5321 \pm .0045$	
Logistic Regression	LDPC	$.9747 \pm .0021$	$.7021 \pm .0049$	$.5295 \pm .0047$	

Table A.2: Macro-F_1 of ML-ECC using 3-powerset base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (RAKEL)	.7653 ± .0019	.6815 ± .0032	.3736 ± .0015	.2538 ± .0034
Random Forest	HAMR	.7840 ± .0018	**.6868 ± .0032**	.3837 ± .0016	.2604 ± .0035
Random Forest	BCH	**.7863 ± .0019**	**.6860 ± .0036**	**.3859 ± .0017**	**.3325 ± .0033**
Random Forest	LDPC	.7666 ± .0021	.6566 ± .0040	.3596 ± .0016	.2845 ± .0031
Gaussian SVM	RREP (RAKEL)	.7955 ± .0013	.5199 ± .0043	**.4252 ± .0018**	**.4586 ± .0047**
Gaussian SVM	HAMR	**.7980 ± .0014**	**.5301 ± .0045**	**.4254 ± .0018**	.4448 ± .0045
Gaussian SVM	BCH	.7940 ± .0016	**.5256 ± .0048**	.4196 ± .0017	.4238 ± .0046
Gaussian SVM	LDPC	.7898 ± .0016	.4409 ± .0078	.4146 ± .0023	.4237 ± .0041
Logistic Regression	RREP (RAKEL)	.7298 ± .0014	**.6456 ± .0043**	.3508 ± .0012	**.4004 ± .0052**
Logistic Regression	HAMR	**.7399 ± .0014**	**.6438 ± .0041**	.3562 ± .0013	.3827 ± .0047
Logistic Regression	BCH	**.7398 ± .0015**	.6267 ± .0042	**.3588 ± .0021**	.3572 ± .0043
Logistic Regression	LDPC	.7179 ± .0019	.6041 ± .0043	.3500 ± .0026	.3777 ± .0046

base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	RREP (RAKEL)	**.7529 ± .0051**	.1370 ± .0030	.1797 ± .0053
Random Forest	HAMR	**.7530 ± .0055**	.1455 ± .0034	.1819 ± .0053
Random Forest	BCH	**.7575 ± .0056**	**.2536 ± .0035**	**.1950 ± .0055**
Random Forest	LDPC	**.7550 ± .0058**	.2104 ± .0030	.1770 ± .0046
Gaussian SVM	RREP (RAKEL)	.7365 ± .0060	.3062 ± .0032	.1329 ± .0025
Gaussian SVM	HAMR	.7347 ± .0060	.3004 ± .0029	.1316 ± .0027
Gaussian SVM	BCH	**.7605 ± .0054**	**.3103 ± .0037**	**.1398 ± .0043**
Gaussian SVM	LDPC	.7530 ± .0061	.2888 ± .0025	.1354 ± .0031
Logistic Regression	RREP (RAKEL)	.6045 ± .0066	.3357 ± .0050	.1330 ± .0024
Logistic Regression	HAMR	.6677 ± .0086	**.3412 ± .0053**	.1279 ± .0020
Logistic Regression	BCH	**.7600 ± .0060**	.3227 ± .0038	**.1410 ± .0038**
Logistic Regression	LDPC	.7494 ± .0075	.3138 ± .0043	**.1386 ± .0030**

Table A.3: Ranking loss of ML-ECC using 3-powerset base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (RAKEL)	.1641 ± .0012	**.2130 ± .0021**	.2439 ± .0012	.2381 ± .0012
Random Forest	HAMR	.1407 ± .0012	**.2107 ± .0023**	**.2378 ± .0011**	.2356 ± .0011
Random Forest	BCH	**.1354 ± .0011**	.2139 ± .0026	.2450 ± .0013	**.2237 ± .0013**
Random Forest	LDPC	.1498 ± .0013	.2291 ± .0028	.2611 ± .0014	.2416 ± .0014
Gaussian SVM	RREP (RAKEL)	.1337 ± .0008	**.3034 ± .0028**	.2299 ± .0010	**.1897 ± .0014**
Gaussian SVM	HAMR	**.1277 ± .0008**	**.3041 ± .0029**	**.2281 ± .0010**	**.1905 ± .0013**
Gaussian SVM	BCH	.1291 ± .0010	.3141 ± .0028	.2333 ± .0009	.1993 ± .0013
Gaussian SVM	LDPC	.1322 ± .0009	.3285 ± .0033	.2395 ± .0012	.1972 ± .0015
Logistic Regression	RREP (RAKEL)	.1779 ± .0010	**.2427 ± .0031**	.2495 ± .0011	**.2020 ± .0016**
Logistic Regression	HAMR	.1660 ± .0010	.2462 ± .0030	**.2468 ± .0010**	.2046 ± .0014
Logistic Regression	BCH	**.1634 ± .0010**	.2613 ± .0030	.2539 ± .0012	.2116 ± .0015
Logistic Regression	LDPC	.1787 ± .0013	.2649 ± .0025	.2572 ± .0015	.2055 ± .0019

base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	RREP (RAKEL)	.0069 ± .0005	.2961 ± .0013	.2745 ± .0031
Random Forest	HAMR	.0067 ± .0005	.2911 ± .0015	.2748 ± .0031
Random Forest	BCH	**.0049 ± .0005**	**.1834 ± .0018**	**.2669 ± .0032**
Random Forest	LDPC	.0062 ± .0004	.2161 ± .0018	.2734 ± .0034
Gaussian SVM	RREP (RAKEL)	.0079 ± .0006	.1301 ± .0018	**.2583 ± .0032**
Gaussian SVM	HAMR	.0077 ± .0007	.1312 ± .0017	**.2585 ± .0034**
Gaussian SVM	BCH	**.0054 ± .0005**	**.1272 ± .0014**	.2668 ± .0037
Gaussian SVM	LDPC	.0070 ± .0005	.1292 ± .0014	**.2612 ± .0036**
Logistic Regression	RREP (RAKEL)	.0111 ± .0005	**.0897 ± .0073**	**.2608 ± .0025**
Logistic Regression	HAMR	.0072 ± .0005	**.0896 ± .0070**	**.2622 ± .0025**
Logistic Regression	BCH	**.0039 ± .0004**	.1185 ± .0028	.2640 ± .0030
Logistic Regression	LDPC	**.0042 ± .0004**	.1041 ± .0043	**.2591 ± .0032**

A.2 ML-ECC With Different Codeword Lengths

Table A.4: $0/1$ loss of ML-ECC on `scene` dataset (3-powerset learners)

base learner	ECC	$M = 31$	$M = 47$	$M = 63$	$M = 95$	$M = 127$
Random Forest	RREP (RAKEL)	$.3303 \pm .0036$	$.3361 \pm .0028$	$.3349 \pm .0026$	$.3365 \pm .0025$	$.3394 \pm .0025$
Random Forest	HAMR	$.3070 \pm .0027$	$\mathbf{.3050 \pm .0028}$	$.2930 \pm .0021$	$\mathbf{.2923 \pm .0024}$	$.2849 \pm .0020$
Random Forest	BCH	$\mathbf{.2822 \pm .0026}$	—	$\mathbf{.2713 \pm .0023}$	—	$\mathbf{.2669 \pm .0020}$
Random Forest	LDPC	$.3141 \pm .0036$	$.3238 \pm .0029$	$.3169 \pm .0029$	$.3196 \pm .0029$	$.3057 \pm .0023$
Gaussian SVM	RREP (RAKEL)	$.2855 \pm .0019$	$.2883 \pm .0019$	$.2860 \pm .0016$	$.2868 \pm .0017$	$.2856 \pm .0016$
Gaussian SVM	HAMR	$.2775 \pm .0019$	$\mathbf{.2743 \pm .0019}$	$.2696 \pm .0016$	$\mathbf{.2683 \pm .0015}$	$.2639 \pm .0017$
Gaussian SVM	BCH	$\mathbf{.2623 \pm .0016}$	—	$\mathbf{.2578 \pm .0019}$	—	$\mathbf{.2576 \pm .0017}$
Gaussian SVM	LDPC	$.2828 \pm .0021$	$.2843 \pm .0023$	$.2823 \pm .0023$	$.2829 \pm .0021$	$.2780 \pm .0020$
Logistic Regression	RREP (RAKEL)	$.3667 \pm .0025$	$.3666 \pm .0022$	$.3612 \pm .0021$	$.3637 \pm .0020$	$.3601 \pm .0019$
Logistic Regression	HAMR	$.3548 \pm .0024$	$\mathbf{.3490 \pm .0025}$	$.3407 \pm .0022$	$\mathbf{.3389 \pm .0017}$	$.3293 \pm .0017$
Logistic Regression	BCH	$\mathbf{.3326 \pm .0027}$	—	$\mathbf{.3196 \pm .0020}$	—	$\mathbf{.3148 \pm .0018}$
Logistic Regression	LDPC	$.3719 \pm .0031$	$.3724 \pm .0030$	$.3690 \pm .0028$	$.3696 \pm .0027$	$.3655 \pm .0028$

Table A.5: Hamming loss of ML-ECC on `scene` dataset (3-powerset learners)

base learner	ECC	$M = 31$	$M = 47$	$M = 63$	$M = 95$	$M = 127$
Random Forest	RREP (RAKEL)	$\mathbf{.0772 \pm .0006}$	$\mathbf{.0762 \pm .0006}$	$\mathbf{.0759 \pm .0006}$	$\mathbf{.0754 \pm .0006}$	$.0755 \pm .0006$
Random Forest	HAMR	$.0800 \pm .0008$	$.0770 \pm .0007$	$\mathbf{.0759 \pm .0005}$	$\mathbf{.0750 \pm .0007}$	$\mathbf{.0746 \pm .0006}$
Random Forest	BCH	$.0798 \pm .0008$	—	$.0768 \pm .0007$	—	$.0753 \pm .0007$
Random Forest	LDPC	$.0825 \pm .0007$	$.0807 \pm .0007$	$.0812 \pm .0008$	$.0801 \pm .0007$	$.0819 \pm .0007$
Gaussian SVM	RREP (RAKEL)	$\mathbf{.0724 \pm .0005}$	$\mathbf{.0721 \pm .0005}$	$\mathbf{.0718 \pm .0004}$	$\mathbf{.0720 \pm .0005}$	$\mathbf{.0719 \pm .0005}$
Gaussian SVM	HAMR	$.0750 \pm .0006$	$.0736 \pm .0006$	$.0730 \pm .0005$	$.0728 \pm .0005$	$.0724 \pm .0005$
Gaussian SVM	BCH	$.0748 \pm .0005$	—	$.0738 \pm .0006$	—	$.0739 \pm .0006$
Gaussian SVM	LDPC	$.0758 \pm .0006$	$.0747 \pm .0006$	$.0754 \pm .0006$	$.0745 \pm .0005$	$.0755 \pm .0006$
Logistic Regression	RREP (RAKEL)	$\mathbf{.0927 \pm .0006}$	$\mathbf{.0920 \pm .0005}$	$\mathbf{.0917 \pm .0006}$	$\mathbf{.0918 \pm .0006}$	$\mathbf{.0915 \pm .0005}$
Logistic Regression	HAMR	$.0962 \pm .0008$	$.0935 \pm .0007$	$.0924 \pm .0006$	$\mathbf{.0918 \pm .0005}$	$\mathbf{.0910 \pm .0005}$
Logistic Regression	BCH	$.0961 \pm .0008$	—	$.0932 \pm .0006$	—	$.0920 \pm .0005$
Logistic Regression	LDPC	$.0992 \pm .0007$	$.0960 \pm .0006$	$.0967 \pm .0005$	$.0962 \pm .0007$	$.0989 \pm .0007$

Table A.6: $0/1$ loss of ML-ECC on `yeast` dataset (3-powerset learners)

base learner	ECC	$M = 63$	$M = 95$	$M = 127$	$M = 191$	$M = 255$
Random Forest	RREP (RAKEL)	$.8130 \pm .0096$	$\mathbf{.7921 \pm .0021}$	$.7931 \pm .0022$	$.7947 \pm .0021$	$.7907 \pm .0023$
Random Forest	HAMR	$.8125 \pm .0024$	$.7949 \pm .0025$	$.7900 \pm .0023$	$\mathbf{.7831 \pm .0021}$	$.7791 \pm .0020$
Random Forest	BCH	$\mathbf{.7853 \pm .0021}$	—	$\mathbf{.7750 \pm .0021}$	—	$\mathbf{.7764 \pm .0020}$
Random Forest	LDPC	$.8081 \pm .0024$	$.8125 \pm .0019$	$.8089 \pm .0022$	$.8178 \pm .0019$	$.8082 \pm .0024$
Gaussian SVM	RREP (RAKEL)	$.7779 \pm .0059$	$\mathbf{.7647 \pm .0023}$	$.7638 \pm .0024$	$.7631 \pm .0023$	$.7601 \pm .0023$
Gaussian SVM	HAMR	$.7752 \pm .0026$	$\mathbf{.7641 \pm .0020}$	$.7600 \pm .0022$	$\mathbf{.7551 \pm .0018}$	$.7522 \pm .0021$
Gaussian SVM	BCH	$\mathbf{.7441 \pm .0021}$	—	$\mathbf{.7409 \pm .0020}$	—	$\mathbf{.7428 \pm .0017}$
Gaussian SVM	LDPC	$.7588 \pm .0024$	$.7684 \pm .0016$	$.7639 \pm .0022$	$.7671 \pm .0020$	$.7574 \pm .0021$
Logistic Regression	RREP (RAKEL)	$.8310 \pm .0049$	$.8202 \pm .0016$	$.8206 \pm .0020$	$.8193 \pm .0021$	$.8161 \pm .0017$
Logistic Regression	HAMR	$.8304 \pm .0022$	$\mathbf{.8158 \pm .0016}$	$.8113 \pm .0020$	$\mathbf{.8067 \pm .0017}$	$.8064 \pm .0018$
Logistic Regression	BCH	$\mathbf{.7953 \pm .0024}$	—	$\mathbf{.7887 \pm .0022}$	—	$\mathbf{.7899 \pm .0020}$
Logistic Regression	LDPC	$.8095 \pm .0027$	$\mathbf{.8140 \pm .0019}$	$.8089 \pm .0026$	$.8160 \pm .0021$	$.8082 \pm .0024$

Table A.7: Hamming loss of ML-ECC on `yeast` dataset (3-powerset learners)

base learner	ECC	$M = 63$	$M = 95$	$M = 127$	$M = 191$	$M = 255$
Random Forest	RREP (RAKEL)	$.1963 \pm .0030$	$\mathbf{.1896 \pm .0008}$	$\mathbf{.1892 \pm .0008}$	$\mathbf{.1887 \pm .0008}$	$\mathbf{.1882 \pm .0008}$
Random Forest	HAMR	$.1992 \pm .0007$	$.1937 \pm .0008$	$.1919 \pm .0007$	$.1900 \pm .0008$	$.1895 \pm .0008$
Random Forest	BCH	$\mathbf{.1952 \pm .0007}$	—	$.1942 \pm .0007$	—	$.1928 \pm .0008$
Random Forest	LDPC	$.1990 \pm .0007$	$.1973 \pm .0007$	$.1996 \pm .0006$	$.1989 \pm .0007$	$.2014 \pm .0007$
Gaussian SVM	RREP (RAKEL)	$\mathbf{.1891 \pm .0016}$	$\mathbf{.1862 \pm .0007}$	$\mathbf{.1862 \pm .0007}$	$\mathbf{.1855 \pm .0006}$	$\mathbf{.1853 \pm .0007}$
Gaussian SVM	HAMR	$.1916 \pm .0007$	$.1888 \pm .0006$	$.1882 \pm .0006$	$.1872 \pm .0006$	$.1867 \pm .0006$
Gaussian SVM	BCH	$\mathbf{.1897 \pm .0008}$	—	$.1907 \pm .0007$	—	$.1898 \pm .0008$
Gaussian SVM	LDPC	$\mathbf{.1901 \pm .0007}$	$.1912 \pm .0007$	$.1911 \pm .0006$	$.1912 \pm .0007$	$.1917 \pm .0007$
Logistic Regression	RREP (RAKEL)	$\mathbf{.2029 \pm .0017}$	$\mathbf{.1999 \pm .0006}$	$\mathbf{.1999 \pm .0007}$	$\mathbf{.1996 \pm .0007}$	$\mathbf{.1993 \pm .0007}$
Logistic Regression	HAMR	$.2064 \pm .0009$	$.2027 \pm .0007$	$.2013 \pm .0007$	$.2004 \pm .0007$	$.2004 \pm .0007$
Logistic Regression	BCH	$.2059 \pm .0008$	—	$.2057 \pm .0008$	—	$.2051 \pm .0008$
Logistic Regression	LDPC	$\mathbf{.2032 \pm .0008}$	$.2025 \pm .0008$	$.2027 \pm .0008$	$.2039 \pm .0007$	$.2054 \pm .0007$

Table A.8: $0/1$ loss of ML-ECC on `emotions` dataset (3-powerset learners)

base learner	ECC	$M = 31$	$M = 47$	$M = 63$	$M = 95$	$M = 127$
Random Forest	RREP (RAKEL)	$\mathbf{.6507 \pm .0050}$	$\mathbf{.6475 \pm .0050}$	$.6455 \pm .0052$	$\mathbf{.6512 \pm .0056}$	$.6465 \pm .0057$
Random Forest	HAMR	$.6597 \pm .0053$	$\mathbf{.6507 \pm .0063}$	$.6512 \pm .0050$	$\mathbf{.6483 \pm .0053}$	$\mathbf{.6373 \pm .0061}$
Random Forest	BCH	$\mathbf{.6502 \pm .0058}$	—	$\mathbf{.6386 \pm .0054}$	—	$\mathbf{.6361 \pm .0057}$
Random Forest	LDPC	$.6637 \pm .0051$	$.6787 \pm .0054$	$.6721 \pm .0051$	$.6769 \pm .0053$	$.6619 \pm .0052$
Gaussian SVM	RREP (RAKEL)	$\mathbf{.7822 \pm .0063}$	$\mathbf{.7818 \pm .0048}$	$\mathbf{.7782 \pm .0057}$	$\mathbf{.7790 \pm .0045}$	$\mathbf{.7759 \pm .0055}$
Gaussian SVM	HAMR	$.7965 \pm .0051$	$\mathbf{.7785 \pm .0063}$	$\mathbf{.7771 \pm .0057}$	$\mathbf{.7769 \pm .0052}$	$\mathbf{.7710 \pm .0054}$
Gaussian SVM	BCH	$.7979 \pm .0058$	—	$\mathbf{.7774 \pm .0057}$	—	$\mathbf{.7744 \pm .0053}$
Gaussian SVM	LDPC	$.8059 \pm .0053$	$.8238 \pm .0060$	$.8104 \pm .0052$	$.8257 \pm .0050$	$.8040 \pm .0044$
Logistic Regression	RREP (RAKEL)	$\mathbf{.7061 \pm .0067}$	$\mathbf{.6977 \pm .0071}$	$\mathbf{.6982 \pm .0066}$	$\mathbf{.6957 \pm .0067}$	$\mathbf{.6949 \pm .0070}$
Logistic Regression	HAMR	$.7188 \pm .0063$	$\mathbf{.7030 \pm .0059}$	$.7071 \pm .0066$	$\mathbf{.7017 \pm .0063}$	$\mathbf{.6964 \pm .0057}$
Logistic Regression	BCH	$.7178 \pm .0057$	—	$.7059 \pm .0046$	—	$.7068 \pm .0046$
Logistic Regression	LDPC	$.7356 \pm .0054$	$.7302 \pm .0072$	$.7287 \pm .0058$	$.7312 \pm .0056$	$.7295 \pm .0056$

Table A.9: Hamming loss of ML-ECC on `emotions` dataset (3-powerset learners)

base learner	ECC	$M = 31$	$M = 47$	$M = 63$	$M = 95$	$M = 127$
Random Forest	RREP (RAKEL)	**.1822 ± .0018**	**.1785 ± .0016**	**.1778 ± .0016**	**.1781 ± .0018**	**.1774 ± .0018**
Random Forest	HAMR	.1889 ± .0020	.1825 ± .0020	.1822 ± .0015	.1814 ± .0019	.1799 ± .0020
Random Forest	BCH	.1928 ± .0022	—	.1856 ± .0018	—	.1859 ± .0021
Random Forest	LDPC	.1887 ± .0019	.1919 ± .0016	.1895 ± .0018	.1909 ± .0017	.1912 ± .0020
Gaussian SVM	RREP (RAKEL)	**.2475 ± .0023**	**.2443 ± .0021**	**.2446 ± .0023**	**.2446 ± .0022**	**.2432 ± .0021**
Gaussian SVM	HAMR	.2627 ± .0023	.2534 ± .0026	.2527 ± .0022	.2497 ± .0023	.2489 ± .0023
Gaussian SVM	BCH	.2734 ± .0025	—	.2669 ± .0026	—	.2644 ± .0019
Gaussian SVM	LDPC	.2635 ± .0024	.2636 ± .0019	.2573 ± .0023	.2625 ± .0024	.2634 ± .0027
Logistic Regression	RREP (RAKEL)	**.2065 ± .0023**	**.2026 ± .0025**	**.2029 ± .0024**	**.2017 ± .0025**	**.2026 ± .0025**
Logistic Regression	HAMR	.2170 ± .0030	.2088 ± .0023	.2110 ± .0025	.2092 ± .0028	.2064 ± .0024
Logistic Regression	BCH	.2279 ± .0021	—	.2243 ± .0025	—	.2233 ± .0022
Logistic Regression	LDPC	.2207 ± .0023	.2150 ± .0027	.2172 ± .0022	.2186 ± .0021	.2202 ± .0021

Table A.10: $0/1$ loss of ML-ECC on `medical` dataset (3-powerset learners)

base learner	ECC	$M = 255$	$M = 383$	$M = 511$	$M = 767$	$M = 1023$
Random Forest	RREP (RAKEL)	.6633 ± .0167	.6435 ± .0030	.6428 ± .0027	.6475 ± .0029	.6496 ± .0024
Random Forest	HAMR	.6369 ± .0029	.6424 ± .0033	.6398 ± .0031	.6406 ± .0021	.6417 ± .0030
Random Forest	BCH	**.4730 ± .0036**	—	**.4697 ± .0034**	—	**.4659 ± .0038**
Random Forest	LDPC	.5237 ± .0036	**.5223 ± .0032**	.5276 ± .0031	**.5250 ± .0032**	.5260 ± .0032
Gaussian SVM	RREP (RAKEL)	.3919 ± .0148	**.3702 ± .0036**	.3684 ± .0034	**.3697 ± .0035**	.3680 ± .0035
Gaussian SVM	HAMR	.3686 ± .0036	**.3672 ± .0036**	.3684 ± .0031	**.3673 ± .0029**	.3640 ± .0031
Gaussian SVM	BCH	**.3422 ± .0028**	—	**.3440 ± .0028**	—	**.3394 ± .0027**
Gaussian SVM	LDPC	.3592 ± .0031	**.3677 ± .0033**	.3752 ± .0036	.3811 ± .0032	.3856 ± .0031
Logistic Regression	RREP (RAKEL)	.5718 ± .0240	.5549 ± .0262	.5508 ± .0252	.5488 ± .0253	.5507 ± .0254
Logistic Regression	HAMR	.5326 ± .0232	.5321 ± .0235	.5310 ± .0229	.5286 ± .0230	.5269 ± .0229
Logistic Regression	BCH	**.3895 ± .0054**	—	**.3871 ± .0051**	—	**.3798 ± .0044**
Logistic Regression	LDPC	.4403 ± .0100	**.4470 ± .0097**	.4563 ± .0094	**.4510 ± .0093**	.4516 ± .0083

Table A.11: Hamming loss of ML-ECC on `medical` dataset (3-powerset learners)

base learner	ECC	$M = 255$	$M = 383$	$M = 511$	$M = 767$	$M = 1023$
Random Forest	RREP (RAKEL)	.0195 ± .0010	.0181 ± .0001	.0181 ± .0001	.0182 ± .0001	.0182 ± .0001
Random Forest	HAMR	.0180 ± .0001	.0181 ± .0001	.0180 ± .0001	**.0180 ± .0001**	.0180 ± .0001
Random Forest	BCH	**.0159 ± .0001**	—	**.0158 ± .0001**	—	**.0158 ± .0001**
Random Forest	LDPC	.0171 ± .0001	**.0177 ± .0001**	.0183 ± .0001	.0189 ± .0001	.0191 ± .0002
Gaussian SVM	RREP (RAKEL)	.0120 ± .0005	.0113 ± .0001	**.0112 ± .0001**	**.0112 ± .0001**	**.0112 ± .0001**
Gaussian SVM	HAMR	**.0112 ± .0001**	**.0112 ± .0001**	**.0112 ± .0001**	**.0112 ± .0001**	**.0111 ± .0001**
Gaussian SVM	BCH	**.0113 ± .0001**	—	.0114 ± .0001	—	.0114 ± .0001
Gaussian SVM	LDPC	.0120 ± .0001	.0126 ± .0001	.0131 ± .0001	.0137 ± .0001	.0140 ± .0001
Logistic Regression	RREP (RAKEL)	.0198 ± .0010	.0193 ± .0012	.0191 ± .0011	.0190 ± .0011	.0190 ± .0011
Logistic Regression	HAMR	.0180 ± .0010	**.0180 ± .0010**	.0179 ± .0010	**.0178 ± .0009**	.0176 ± .0009
Logistic Regression	BCH	**.0139 ± .0003**	—	**.0137 ± .0003**	—	**.0137 ± .0003**
Logistic Regression	LDPC	.0167 ± .0006	**.0175 ± .0006**	.0182 ± .0007	**.0184 ± .0007**	.0187 ± .0006

A.3 ML-ECC Using Binary Relevance Base Learners

Table A.12: Micro-F_1 of ML-ECC using BR base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (BR)	.7058 ± .0018	.6817 ± .0034	.6397 ± .0013	.6044 ± .0017
Random Forest	HAMR	.7729 ± .0016	.6991 ± .0032	**.6575 ± .0014**	.6156 ± .0015
Random Forest	BCH	**.7889 ± .0020**	**.7033 ± .0028**	.6533 ± .0015	**.6319 ± .0016**
Random Forest	LDPC	.7276 ± .0022	.6734 ± .0031	.6350 ± .0012	.5882 ± .0019
Gaussian SVM	RREP (BR)	.7749 ± .0014	.5068 ± .0044	.6646 ± .0013	**.6582 ± .0018**
Gaussian SVM	HAMR	**.7881 ± .0012**	.5370 ± .0039	**.6713 ± .0014**	**.6578 ± .0019**
Gaussian SVM	BCH	.7866 ± .0015	**.5538 ± .0052**	.6687 ± .0013	**.6587 ± .0018**
Gaussian SVM	LDPC	.7781 ± .0016	.5047 ± .0047	.6598 ± .0012	.6332 ± .0017
Logistic Regression	RREP (BR)	.6775 ± .0020	.6344 ± .0036	.6349 ± .0015	.6475 ± .0021
Logistic Regression	HAMR	.7076 ± .0017	**.6507 ± .0039**	**.6406 ± .0015**	.6466 ± .0020
Logistic Regression	BCH	**.7212 ± .0018**	**.6499 ± .0043**	.6374 ± .0016	**.6496 ± .0019**
Logistic Regression	LDPC	.6772 ± .0023	.6278 ± .0039	.6296 ± .0017	.6257 ± .0023
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (BR)	.9864 ± .0010	.5432 ± .0023	**.5454 ± .0046**	
Random Forest	HAMR	.9872 ± .0009	.5591 ± .0024	**.5472 ± .0046**	
Random Forest	BCH	**.9892 ± .0009**	**.6869 ± .0032**	.5399 ± .0042	
Random Forest	LDPC	.9844 ± .0012	.5812 ± .0028	.5201 ± .0047	
Gaussian SVM	RREP (BR)	.9874 ± .0010	**.7818 ± .0022**	**.5696 ± .0053**	
Gaussian SVM	HAMR	**.9894 ± .0010**	**.7831 ± .0021**	**.5664 ± .0050**	
Gaussian SVM	BCH	**.9892 ± .0008**	.7728 ± .0019	.5466 ± .0058	
Gaussian SVM	LDPC	.9878 ± .0008	.7249 ± .0025	.5349 ± .0052	
Logistic Regression	RREP (BR)	.7272 ± .0035	.6935 ± .0114	**.5713 ± .0034**	
Logistic Regression	HAMR	.8326 ± .0040	.7118 ± .0091	**.5682 ± .0035**	
Logistic Regression	BCH	**.9752 ± .0017**	**.7284 ± .0057**	.5624 ± .0044	
Logistic Regression	LDPC	.9492 ± .0033	.6612 ± .0077	.5397 ± .0036	

Table A.13: Macro-F_1 of ML-ECC using BR base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (BR)	.7045 ± .0016	.6513 ± .0037	.3554 ± .0012	.2596 ± .0035
Random Forest	HAMR	.7758 ± .0016	.6738 ± .0036	.3802 ± .0014	.2836 ± .0035
Random Forest	BCH	**.7950 ± .0019**	**.6863 ± .0030**	**.3845 ± .0017**	**.3528 ± .0031**
Random Forest	LDPC	.7280 ± .0020	.6449 ± .0034	.3676 ± .0011	.2872 ± .0032
Gaussian SVM	RREP (BR)	.7810 ± .0013	.4362 ± .0046	.4138 ± .0018	**.4560 ± .0044**
Gaussian SVM	HAMR	**.7950 ± .0012**	.4780 ± .0043	**.4188 ± .0019**	.4262 ± .0044
Gaussian SVM	BCH	**.7940 ± .0016**	**.5190 ± .0052**	**.4198 ± .0020**	.4484 ± .0043
Gaussian SVM	LDPC	.7845 ± .0015	.4420 ± .0048	.4161 ± .0019	.4165 ± .0036
Logistic Regression	RREP (BR)	.6817 ± .0019	.6062 ± .0044	.3405 ± .0016	**.4436 ± .0044**
Logistic Regression	HAMR	.7119 ± .0015	.6267 ± .0042	.3445 ± .0014	.4041 ± .0048
Logistic Regression	BCH	**.7264 ± .0016**	**.6336 ± .0042**	**.3482 ± .0020**	.4306 ± .0044
Logistic Regression	LDPC	.6803 ± .0024	.5969 ± .0050	**.3466 ± .0016**	.4050 ± .0043
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (BR)	.7527 ± .0052	.1338 ± .0026	.1794 ± .0052	
Random Forest	HAMR	**.7535 ± .0053**	.1447 ± .0029	.1822 ± .0054	
Random Forest	BCH	**.7589 ± .0057**	**.2628 ± .0036**	**.1945 ± .0053**	
Random Forest	LDPC	.7499 ± .0055	.1963 ± .0027	.1777 ± .0046	
Gaussian SVM	RREP (BR)	.7659 ± .0065	**.3081 ± .0034**	.1414 ± .0029	
Gaussian SVM	HAMR	.7628 ± .0058	.2958 ± .0028	.1359 ± .0029	
Gaussian SVM	BCH	.7612 ± .0052	**.3092 ± .0040**	**.1706 ± .0049**	
Gaussian SVM	LDPC	**.7731 ± .0060**	.2880 ± .0025	.1476 ± .0032	
Logistic Regression	RREP (BR)	.5425 ± .0043	.3252 ± .0042	.1478 ± .0027	
Logistic Regression	HAMR	.6052 ± .0057	**.3314 ± .0057**	.1375 ± .0028	
Logistic Regression	BCH	**.7401 ± .0065**	.3137 ± .0041	**.1672 ± .0037**	
Logistic Regression	LDPC	.7183 ± .0081	.3001 ± .0040	.1459 ± .0025	

Table A.14: Ranking loss of ML-ECC using BR base learners

base learner	ECC	scene (M=127)	emotions (M=127)	yeast (M=255)	tmc2007 (M=511)
Random Forest	RREP (BR)	.2113 ± .0010	.2336 ± .0023	.2559 ± .0010	.2357 ± .0012
Random Forest	HAMR	.1554 ± .0010	.2185 ± .0022	**.2422 ± .0011**	.2275 ± .0011
Random Forest	BCH	**.1294 ± .0012**	**.2139 ± .0019**	.2461 ± .0011	**.2148 ± .0012**
Random Forest	LDPC	.1907 ± .0013	.2397 ± .0022	.2585 ± .0010	.2351 ± .0013
Gaussian SVM	RREP (BR)	.1493 ± .0008	.3375 ± .0025	.2379 ± .0010	**.1914 ± .0013**
Gaussian SVM	HAMR	.1348 ± .0007	.3231 ± .0024	**.2321 ± .0011**	.1937 ± .0014
Gaussian SVM	BCH	**.1286 ± .0010**	**.3179 ± .0034**	.2340 ± .0010	**.1923 ± .0013**
Gaussian SVM	LDPC	.1435 ± .0010	.3406 ± .0024	.2411 ± .0010	.1973 ± .0013
Logistic Regression	RREP (BR)	.2128 ± .0013	.2627 ± .0025	.2571 ± .0012	**.1977 ± .0016**
Logistic Regression	HAMR	.1890 ± .0011	**.2515 ± .0028**	**.2523 ± .0012**	.2006 ± .0015
Logistic Regression	BCH	**.1700 ± .0011**	.2543 ± .0030	.2559 ± .0013	**.1965 ± .0015**
Logistic Regression	LDPC	.2100 ± .0016	.2673 ± .0026	.2605 ± .0013	.2012 ± .0018
base learner	ECC	genbase (M=511)	medical (M=1023)	enron (M=1023)	
Random Forest	RREP (BR)	.0075 ± .0006	.2990 ± .0014	.2715 ± .0029	
Random Forest	HAMR	.0068 ± .0004	.2900 ± .0014	.2695 ± .0029	
Random Forest	BCH	**.0048 ± .0005**	**.1844 ± .0018**	**.2660 ± .0029**	
Random Forest	LDPC	.0067 ± .0004	.2365 ± .0017	.2713 ± .0030	
Gaussian SVM	RREP (BR)	.0065 ± .0005	.1297 ± .0019	**.2574 ± .0032**	
Gaussian SVM	HAMR	.0058 ± .0005	.1307 ± .0018	**.2596 ± .0032**	
Gaussian SVM	BCH	**.0049 ± .0004**	**.1265 ± .0012**	**.2600 ± .0034**	
Gaussian SVM	LDPC	.0054 ± .0005	.1311 ± .0018	**.2599 ± .0033**	
Logistic Regression	RREP (BR)	.0199 ± .0005	**.0913 ± .0075**	.2570 ± .0024	
Logistic Regression	HAMR	.0116 ± .0005	**.0909 ± .0075**	.2601 ± .0024	
Logistic Regression	BCH	**.0041 ± .0004**	.1074 ± .0039	**.2528 ± .0029**	
Logistic Regression	LDPC	.0048 ± .0005	.0994 ± .0063	.2562 ± .0025	

A.4 ML-ECC With Geometric Decoders

Table A.15: Micro-F_1 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)
Random Forest	HAMR	alg-hard	.7733 ± .0016	.6992 ± .0031	.6575 ± .0014	**.6156 ± .0015**
Random Forest	HAMR	geo-hard	**.7776 ± .0017**	**.7061 ± .0031**	**.6614 ± .0015**	**.6162 ± .0016**
Random Forest	HAMR	geo-soft	.7710 ± .0018	.7023 ± .0031	.6553 ± .0014	**.6159 ± .0016**
Gaussian SVM	HAMR	alg-hard	.7879 ± .0012	.5375 ± .0040	**.6713 ± .0014**	.6578 ± .0019
Gaussian SVM	HAMR	geo-hard	.7890 ± .0013	**.5502 ± .0042**	**.6726 ± .0014**	**.6584 ± .0019**
Gaussian SVM	HAMR	geo-soft	**.7929 ± .0012**	.5305 ± .0041	**.6714 ± .0014**	**.6598 ± .0019**
Logistic Regression	HAMR	alg-hard	.7076 ± .0017	**.6511 ± .0040**	**.6406 ± .0015**	.6466 ± .0020
Logistic Regression	HAMR	geo-hard	.7096 ± .0018	**.6549 ± .0041**	**.6413 ± .0015**	**.6470 ± .0020**
Logistic Regression	HAMR	geo-soft	**.7156 ± .0019**	**.6544 ± .0045**	**.6417 ± .0014**	**.6486 ± .0020**
Random Forest	BCH	alg-hard	.7897 ± .0018	.7032 ± .0028	.6533 ± .0015	.6319 ± .0016
Random Forest	BCH	geo-hard	.7897 ± .0018	.7059 ± .0027	.6658 ± .0017	.6490 ± .0018
Random Forest	BCH	geo-soft	**.7944 ± .0019**	**.7153 ± .0025**	**.6731 ± .0014**	**.6582 ± .0016**
Gaussian SVM	BCH	alg-hard	.7864 ± .0016	.5531 ± .0051	.6687 ± .0013	**.6587 ± .0018**
Gaussian SVM	BCH	geo-hard	.7886 ± .0015	.5834 ± .0047	.6693 ± .0012	.6426 ± .0020
Gaussian SVM	BCH	geo-soft	**.7924 ± .0016**	**.5902 ± .0046**	**.6755 ± .0012**	**.6590 ± .0018**
Logistic Regression	BCH	alg-hard	.7212 ± .0018	.6499 ± .0043	.6374 ± .0016	**.6496 ± .0019**
Logistic Regression	BCH	geo-hard	.7236 ± .0017	.6471 ± .0044	.6395 ± .0013	.6299 ± .0022
Logistic Regression	BCH	geo-soft	**.7337 ± .0016**	**.6652 ± .0043**	**.6462 ± .0013**	**.6480 ± .0017**

base learner	ECC	decoder	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	HAMR	alg-hard	**.9872 ± .0009**	**.5591 ± .0024**	**.5472 ± .0046**
Random Forest	HAMR	geo-hard	**.9875 ± .0009**	**.5604 ± .0023**	**.5477 ± .0045**
Random Forest	HAMR	geo-soft	**.9878 ± .0009**	**.5603 ± .0024**	**.5467 ± .0047**
Gaussian SVM	HAMR	alg-hard	**.9894 ± .0010**	**.7831 ± .0021**	**.5664 ± .0050**
Gaussian SVM	HAMR	geo-hard	**.9900 ± .0010**	**.7831 ± .0021**	**.5668 ± .0052**
Gaussian SVM	HAMR	geo-soft	**.9897 ± .0010**	**.7848 ± .0021**	**.5659 ± .0053**
Logistic Regression	HAMR	alg-hard	.8326 ± .0040	.7118 ± .0091	**.5682 ± .0035**
Logistic Regression	HAMR	geo-hard	.8324 ± .0043	.7107 ± .0093	**.5677 ± .0035**
Logistic Regression	HAMR	geo-soft	**.8742 ± .0039**	**.7228 ± .0080**	**.5683 ± .0036**
Random Forest	BCH	alg-hard	**.9892 ± .0009**	.6869 ± .0032	**.5399 ± .0042**
Random Forest	BCH	geo-hard	**.9889 ± .0009**	.7048 ± .0028	.5283 ± .0041
Random Forest	BCH	geo-soft	**.9891 ± .0009**	**.7106 ± .0030**	.5337 ± .0044
Gaussian SVM	BCH	alg-hard	**.9892 ± .0008**	**.7728 ± .0019**	**.5466 ± .0058**
Gaussian SVM	BCH	geo-hard	**.9887 ± .0008**	.7603 ± .0026	.5128 ± .0053
Gaussian SVM	BCH	geo-soft	**.9887 ± .0009**	.7689 ± .0026	.5155 ± .0057
Logistic Regression	BCH	alg-hard	**.9752 ± .0017**	.7284 ± .0057	**.5624 ± .0044**
Logistic Regression	BCH	geo-hard	.9451 ± .0041	.7260 ± .0053	.5250 ± .0041
Logistic Regression	BCH	geo-soft	.9532 ± .0031	**.7439 ± .0036**	.5251 ± .0043

Table A.16: Macro-F_1 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)
Random Forest	HAMR	alg-hard	.7760 ± .0016	.6741 ± .0035	.3802 ± .0014	**.2836 ± .0035**
Random Forest	HAMR	geo-hard	**.7808 ± .0017**	**.6823 ± .0034**	**.3853 ± .0015**	**.2837 ± .0037**
Random Forest	HAMR	geo-soft	.7741 ± .0017	.6756 ± .0034	.3759 ± .0014	**.2817 ± .0035**
Gaussian SVM	HAMR	alg-hard	.7949 ± .0012	.4783 ± .0042	**.4188 ± .0019**	**.4262 ± .0044**
Gaussian SVM	HAMR	geo-hard	.7961 ± .0013	**.4940 ± .0051**	**.4198 ± .0018**	**.4269 ± .0045**
Gaussian SVM	HAMR	geo-soft	**.8004 ± .0013**	.4624 ± .0042	.4177 ± .0017	**.4281 ± .0042**
Logistic Regression	HAMR	alg-hard	.7118 ± .0016	.6269 ± .0043	**.3445 ± .0014**	.4041 ± .0048
Logistic Regression	HAMR	geo-hard	.7139 ± .0015	**.6320 ± .0043**	**.3450 ± .0013**	.4062 ± .0051
Logistic Regression	HAMR	geo-soft	**.7204 ± .0017**	**.6310 ± .0047**	**.3447 ± .0012**	**.4150 ± .0046**
Random Forest	BCH	alg-hard	.7957 ± .0018	.6860 ± .0029	.3845 ± .0017	.3528 ± .0031
Random Forest	BCH	geo-hard	.7961 ± .0018	.6910 ± .0029	**.4166 ± .0018**	**.3883 ± .0032**
Random Forest	BCH	geo-soft	**.8006 ± .0018**	**.6975 ± .0025**	.4120 ± .0016	.3788 ± .0034
Gaussian SVM	BCH	alg-hard	.7938 ± .0016	.5181 ± .0050	.4198 ± .0020	**.4484 ± .0043**
Gaussian SVM	BCH	geo-hard	.7961 ± .0016	**.5582 ± .0048**	**.4312 ± .0019**	.3851 ± .0035
Gaussian SVM	BCH	geo-soft	**.8003 ± .0016**	**.5607 ± .0046**	**.4300 ± .0019**	.3957 ± .0038
Logistic Regression	BCH	alg-hard	.7264 ± .0016	.6336 ± .0042	.3482 ± .0020	**.4306 ± .0044**
Logistic Regression	BCH	geo-hard	.7289 ± .0015	.6322 ± .0043	**.3583 ± .0018**	.3537 ± .0035
Logistic Regression	BCH	geo-soft	**.7397 ± .0014**	**.6499 ± .0044**	**.3588 ± .0017**	.3695 ± .0035

base learner	ECC	decoder	genbase (M=511)	medical (M=1023)	enron (M=1023)
Random Forest	HAMR	alg-hard	**.7535 ± .0053**	**.1447 ± .0029**	**.1822 ± .0054**
Random Forest	HAMR	geo-hard	**.7550 ± .0053**	**.1448 ± .0028**	**.1836 ± .0054**
Random Forest	HAMR	geo-soft	**.7556 ± .0052**	**.1449 ± .0027**	**.1830 ± .0053**
Gaussian SVM	HAMR	alg-hard	.7628 ± .0058	**.2958 ± .0028**	**.1359 ± .0029**
Gaussian SVM	HAMR	geo-hard	**.7704 ± .0068**	**.2943 ± .0028**	**.1358 ± .0029**
Gaussian SVM	HAMR	geo-soft	**.7685 ± .0069**	**.2932 ± .0031**	.1325 ± .0027
Logistic Regression	HAMR	alg-hard	.6052 ± .0057	**.3314 ± .0057**	.1375 ± .0028
Logistic Regression	HAMR	geo-hard	.6074 ± .0063	.3305 ± .0052	.1368 ± .0027
Logistic Regression	HAMR	geo-soft	**.6375 ± .0072**	**.3371 ± .0059**	**.1417 ± .0024**
Random Forest	BCH	alg-hard	**.7589 ± .0057**	.2628 ± .0036	.1945 ± .0053
Random Forest	BCH	geo-hard	**.7580 ± .0057**	**.2878 ± .0038**	.1965 ± .0040
Random Forest	BCH	geo-soft	**.7579 ± .0053**	**.2860 ± .0039**	**.2041 ± .0051**
Gaussian SVM	BCH	alg-hard	**.7612 ± .0052**	.3092 ± .0040	**.1706 ± .0049**
Gaussian SVM	BCH	geo-hard	**.7609 ± .0054**	**.3227 ± .0044**	**.1659 ± .0035**
Gaussian SVM	BCH	geo-soft	**.7609 ± .0053**	.3174 ± .0043	**.1675 ± .0045**
Logistic Regression	BCH	alg-hard	**.7401 ± .0065**	.3137 ± .0041	.1672 ± .0037
Logistic Regression	BCH	geo-hard	.6968 ± .0082	.3091 ± .0035	.1702 ± .0032
Logistic Regression	BCH	geo-soft	.7033 ± .0079	**.3235 ± .0037**	**.1833 ± .0040**

Table A.17: Ranking 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	$.1553 \pm .0010$	$.2183 \pm .0021$	$.2422 \pm .0011$	$.2275 \pm .0011$	$.0068 \pm .0004$	$.2900 \pm .0014$	$.2695 \pm .0029$
Random Forest	HAMR	geo-hard	$.0760 \pm .0010$	$.1707 \pm .0026$	$.1820 \pm .0012$	$.1068 \pm .0010$	$.0038 \pm .0004$	$.1265 \pm .0020$	$.1762 \pm .0019$
Random Forest	HAMR	geo-soft	$\mathbf{.0556 \pm .0007}$	$\mathbf{.1369 \pm .0021}$	$\mathbf{.1569 \pm .0010}$	$\mathbf{.0487 \pm .0005}$	$\mathbf{.0021 \pm .0003}$	$\mathbf{.0305 \pm .0009}$	$\mathbf{.0765 \pm .0008}$
Gaussian SVM	HAMR	alg-hard	$.1351 \pm .0007$	$.3226 \pm .0025$	$.2321 \pm .0011$	$.1937 \pm .0014$	$.0058 \pm .0005$	$.1307 \pm .0018$	$.2596 \pm .0032$
Gaussian SVM	HAMR	geo-hard	$.0913 \pm .0012$	$.2685 \pm .0035$	$.1835 \pm .0011$	$.0918 \pm .0007$	$\mathbf{.0028 \pm .0004}$	$.0656 \pm .0014$	$.1657 \pm .0024$
Gaussian SVM	HAMR	geo-soft	$\mathbf{.0571 \pm .0008}$	$\mathbf{.2307 \pm .0029}$	$\mathbf{.1547 \pm .0010}$	$\mathbf{.0514 \pm .0005}$	$\mathbf{.0030 \pm .0006}$	$\mathbf{.0290 \pm .0009}$	$\mathbf{.0828 \pm .0009}$
Logistic Regression	HAMR	alg-hard	$.1890 \pm .0011$	$.2513 \pm .0029$	$.2523 \pm .0012$	$.2006 \pm .0015$	$.0116 \pm .0005$	$.0909 \pm .0075$	$.2601 \pm .0024$
Logistic Regression	HAMR	geo-hard	$.1227 \pm .0015$	$.2047 \pm .0033$	$.1990 \pm .0012$	$.0935 \pm .0009$	$\mathbf{.0028 \pm .0004}$	$.0451 \pm .0033$	$.1554 \pm .0020$
Logistic Regression	HAMR	geo-soft	$\mathbf{.0810 \pm .0007}$	$\mathbf{.1574 \pm .0026}$	$\mathbf{.1689 \pm .0010}$	$\mathbf{.0513 \pm .0005}$	$\mathbf{.0032 \pm .0005}$	$\mathbf{.0264 \pm .0008}$	$\mathbf{.0811 \pm .0011}$
Random Forest	BCH	alg-hard	$.1289 \pm .0011$	$.2140 \pm .0019$	$.2461 \pm .0011$	$.2148 \pm .0012$	$.0048 \pm .0005$	$.1844 \pm .0018$	$.2660 \pm .0029$
Random Forest	BCH	geo-hard	$.0977 \pm .0011$	$.1812 \pm .0027$	$.2181 \pm .0013$	$.1275 \pm .0009$	$.0038 \pm .0004$	$.1227 \pm .0016$	$.2254 \pm .0022$
Random Forest	BCH	geo-soft	$\mathbf{.0570 \pm .0007}$	$\mathbf{.1437 \pm .0021}$	$\mathbf{.1737 \pm .0012}$	$\mathbf{.0602 \pm .0006}$	$\mathbf{.0032 \pm .0005}$	$\mathbf{.0718 \pm .0023}$	$\mathbf{.1465 \pm .0014}$
Gaussian SVM	BCH	alg-hard	$.1287 \pm .0010$	$.3187 \pm .0033$	$.2340 \pm .0010$	$.1923 \pm .0013$	$.0049 \pm .0004$	$.1265 \pm .0012$	$.2600 \pm .0034$
Gaussian SVM	BCH	geo-hard	$.1056 \pm .0011$	$.2890 \pm .0038$	$.2164 \pm .0012$	$.1251 \pm .0010$	$\mathbf{.0034 \pm .0005}$	$.0863 \pm .0014$	$.2308 \pm .0030$
Gaussian SVM	BCH	geo-soft	$\mathbf{.0685 \pm .0009}$	$\mathbf{.2419 \pm .0035}$	$\mathbf{.1767 \pm .0012}$	$\mathbf{.0674 \pm .0007}$	$\mathbf{.0034 \pm .0006}$	$\mathbf{.0652 \pm .0021}$	$\mathbf{.1435 \pm .0017}$
Logistic Regression	BCH	alg-hard	$.1700 \pm .0011$	$.2543 \pm .0030$	$.2559 \pm .0013$	$.1965 \pm .0015$	$.0041 \pm .0004$	$.1074 \pm .0039$	$.2528 \pm .0029$
Logistic Regression	BCH	geo-hard	$.1361 \pm .0013$	$.2293 \pm .0040$	$.2360 \pm .0011$	$.1320 \pm .0011$	$.0049 \pm .0006$	$.0848 \pm .0021$	$.2251 \pm .0022$
Logistic Regression	BCH	geo-soft	$\mathbf{.0884 \pm .0008}$	$\mathbf{.1793 \pm .0033}$	$\mathbf{.1924 \pm .0013}$	$\mathbf{.0669 \pm .0007}$	$\mathbf{.0026 \pm .0004}$	$\mathbf{.0672 \pm .0023}$	$\mathbf{.1786 \pm .0019}$