Linear Batch Codes

Helger Lipmaa and Vitaly Skachek

Abstract In an application, where a client wants to obtain many symbols from a large database, it is often desirable to balance the load. Batch codes (introduced by Ishai *et al.* in STOC 2004) do exactly that: the large database is divided between many servers, so that the client has to only make a small number of queries to every server to obtain sufficient information to reconstruct all desired symbols.

In this work, we formalize the study of *linear* batch codes. These codes, in particular, are of potential use in distributed storage systems. We show that a generator matrix of a binary linear batch code is also a generator matrix of classical binary linear error-correcting code. This immediately yields that a variety of upper bounds, which were developed for error-correcting codes, are applicable also to binary linear batch codes. We also propose new methods for constructing large linear batch codes from the smaller ones.

Key words: Batch codes, error-correcting codes, computationally-private information retrieval, load balancing, distributed storage.

1 Introduction

Consider the scenario where a client wants to retrieve m symbols from an n symbol database. Accessing a single server by all clients simultaneously can create serious performance problems. A simple solution is to replicate the whole database between M servers, so that the client can query approximately m/M symbols from every server. However, that solution require to store $N = Mn$ database symbols.

In an m-out-of-n CPIR (computationally-private information retrieval [8]), the client wants to retrieve m symbols from an n symbol database without the storage

Helger Lipmaa and Vitaly Skachek
Institute of Computer Science, University of Tartu, J. Liivi 2, Tartu 50409, Estonia
e-mail: helger.lipmaa@ut.ee, vitaly.skachek@ut.ee

provider getting to know which symbols were retrieved. An additional problem in this case is the storage provider's computational complexity that is $\Theta(n)$ per query in almost all known 1-out-of-n CPIR protocols. (The only exception is [9], where the per-query computational complexity is $O(n/\log n)$.) Just performing m instances of an 1-out-of-n CPIR protocol would result in a highly prohibitive computational complexity.

To tackle both mentioned problems, Ishai *et al.* [7] proposed to use *batch codes*. More precisely, let Σ be a finite alphabet. In an $(n,N,m,M,T)_\Sigma$ batch code, a database f of n strings in Σ is divided into M buckets where each bucket contains N/M strings in Σ. If a client is to obtain m symbols of the original database, he query (no more than) T symbols from each of the M buckets.

Batch codes have been recently studied very actively in the combinatorial setting. Namely, a *combinatorial batch code* (CBC) satisfies the additional requirement that every symbol of every bucket is equal to some symbol of the original database. (See for example [3, 2, 4].) New constructions of combinatorial batch codes, based on affine planes and transversal designs, were recently presented in [15].

We stress that linear batch codes are also well suitable for the use in the *distributed data storage* [6]. The buckets can be viewed as servers. The reading of the requested data can be done "locally" from a small number of servers. If a small number of buckets stopped functioning, the data can be reproduced by reading data from (a small number) of other buckets.

In this paper, we formalize a framework for analysis of linear batch codes, which resembles that of the classical error-correcting codes (ECCs). As we show, generator matrices of good binary linear batch codes are also generator matrices of good classical ECCs. This immediately gives us a set of tools and bounds from the classical coding theory for analyzing binary linear batch codes. The converse, however, is not true: not every good binary linear ECC is a good linear batch code. Finally, we present a number of simple constructions of larger linear batch codes from the smaller ones. The preliminary version of this paper is available as [10].

2 Preliminaries

Let $[n] \triangleq \{1,2,\cdots,n\}$. We denote by $\langle v_i \rangle_{i\in[n]}$ the linear span of the vectors v_i, $i \in [n]$, over some finite field $\mathbb{F}_q$. We use notation $\mathrm{d}_H(x,y)$ to denote the Hamming distance between the vectors x and y, and notation $\mathrm{w}_H(x)$ to denote the Hamming weight of x. We also denote by 0 the row vector consisting of all zeros, and by e_i the row vector having one at position i and zeros elsewhere (the length of vectors will be clear from the context).

We start with the definition of a batch code. In this work, we focus on so-called *multiset* batch codes, as they were defined in [7].

Definition 1 ([7]). Let Σ be a finite alphabet. We say that $\mathscr{C}$ is an $(n,N,m,M,t)_\Sigma$ batch code over a finite alphabet Σ if it encodes any string $x = (x_1,x_2,\cdots,x_n) \in \Sigma^n$ into M strings (buckets) of total length N over Σ, namely $y_1,y_2,\cdots,y_M$, such that for each m-tuple (batch) of (not neccessarily distinct) indices $i_1, i_2, \cdots, i_m \in [n]$, the symbols $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ can be retrieved by m users, respectively, by reading at most t symbols from each bucket, such that each symbol x_{i_ℓ} is recovered from the symbols read by the ℓ-th user alone.

The ratio $R \overset{\triangle}{=} n/N$ is called the rate of the code.

If for the code $\mathscr{C}$ it holds that $t = 1$, then we use notation $(n,N,m,M)_\Sigma$ for it. This corresponds to an important special case when only one symbol is read from each bucket. Note that the buckets in this definition correspond to the devices in the above example, the encoding length N to the total storage, and the parameter t to the maximal load. If $\Sigma = \mathbb{F}_q$ is a finite field, we also use notation $(n,N,m,M,t)_q$ (or $(n,N,m,M)_q$) to denote $(n,N,m,M,t)_\Sigma$ (or $(n,N,m,M)_\Sigma$, respectively).

Definition 2. We say that an $(n,N,m,M,t)_q$ batch code is *linear*, if every symbol in every bucket is a linear combination of original database symbols.

3 Linear batch codes

In what follows, we consider the case of a linear batch code $\mathscr{C}$ with $t = 1$. Moreover, we limit ourselves to the case when $N = M$, which means that each encoded bucket contains just one symbol in $\mathbb{F}_q$.

Definition 3. For simplicity we refer to a linear $(n,N = M,m,M)_q$ batch code as $[M,n,m]_q$ batch code.

As before, let $x = (x_1, x_2, \cdots, x_n)$ be an information string, and let $y = (y_1, y_2, \cdots, y_M)$ be an encoding of x. Due to linearity of the code, each encoded symbol y_i, $i \in [M]$, can be written as $y_i = \sum_{j=1}^n g_{j,i} x_j$ for some symbols $g_{j,i} \in \mathbb{F}_q$, $j \in [n]$, $i \in [M]$. Then we can form the matrix G as follows:

$$G = \Big(g_{j,i} \Big)_{j \in [n], i \in [M]} ,$$

and thus the encoding is $y = xG$.

The $n \times M$ binary matrix G play a role similar to a generator matrix for a classical linear ECC. In the sequel, we call G a *generator matrix* of the batch code $\mathscr{C}$. We denote by G_i the i-th row of G and by $G^{[\ell]}$ the ℓ-th column of G.

Theorem 1. *Let $\mathscr{C}$ be an $[M,n,m]_q$ batch code. It is possible to retrieve $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ simultaneously if and only if there exist m non-intersecting sets $T_1, T_2, \cdots, T_m$ of indices of columns in G, and for T_r there exists a linear combination of columns of G indexed by that set, which equals to the column vector $e_{i_r}^T$, for all $r \in [m]$.*

Proof. 1. For each $r \in [m]$

$$e_{i_r}^T = \sum_{\ell \in T_r} \alpha_\ell \cdot G^{[\ell]} ,$$

where all $\alpha_\ell \in \mathbb{F}_q$. Due to linearity, the encoding of $x = (x_1, x_2, \cdots, x_n)$ can be written as $y = (y_1, y_2, \cdots, y_M) = x \cdot G$. Then,

$$x_{i_r} = x \cdot e_{i_r}^T = x \cdot \left(\sum_{\ell \in T_r} \alpha_\ell \cdot G^{[\ell]} \right) = \sum_{\ell \in T_r} \alpha_\ell (x \cdot G^{[\ell]}) = \sum_{\ell \in T_r} \alpha_\ell \cdot y_\ell \; ,$$

and therefore the value of x_{i_r} can be obtained by querying only the values of y_ℓ for $\ell \in T_r$. The conclusion follows from the fact that all T_r do not intersect.

2. To show the opposite direction of Theorem 1, we follow the idea of the proof of Theorem 1 in [1]. Let T_r, for $r \in [m]$, be a set of indices of entries in y, which are used to retrieve x_{i_r}. We show that $e_{i_r}^T \in \langle G^{[\ell]} \rangle_{\ell \in T_r}$.
Denote the vector space $W_r \triangleq \langle G^{[\ell]} \rangle_{\ell \in T_r}$. Assume by contradiction that $e_{i_r} \notin W_r$. Recall that the dual space of W_r, denoted by $W_r^\perp$, consists of all the vectors orthogonal to any vector in W_r. Since $e_{i_r} \notin W_r$, there exists a vector $z \in W_r^\perp$, which is not orthogonal to e_{i_r}, i.e. $z \cdot e_{i_r} \neq 0$, and so $z_{i_r} \neq 0$. On the other hand, this vector z is orthogonal to any vector $G^{[\ell]}$ for $\ell \in T_r$.
Consider the encoding of the vectors z and 0, $z \cdot G$ and $0 \cdot G$, respectively. In both cases, all the coordinates of y indexed by T_r are all zeros. Therefore, the result of the retrieval of the i_r-th encoded symbol in both cases is the same, yet $z_{i_r} \neq 0$. We obtain a contradiction.

Example 1. Consider the following linear binary batch code $\mathscr{C}$ whose 4×9 generator matrix is given by

$$G = \begin{pmatrix} 1&0&1&0&0&0&1&0&1 \\ 0&1&1&0&0&0&0&1&1 \\ 0&0&0&1&0&1&1&0&1 \\ 0&0&0&0&1&1&0&1&1 \end{pmatrix} .$$

Let $x = (x_1, x_2, x_3, x_4)$, $y = xG$.

Assume that we want to retrieve the values of (x_1, x_1, x_2, x_2). We can retrieve (x_1, x_1, x_2, x_2) from the following set of equations:

$$\begin{cases} x_1 = y_1 \\ x_1 = y_2 + y_3 \\ x_2 = y_5 + y_8 \\ x_2 = y_4 + y_6 + y_7 + y_9 \end{cases} .$$

Moreover, it is straightforward to verify that any 4-tuple $(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4})$, where $i_1, i_2, i_3, i_4 \in [4]$, can be retrieved by using columns indexed by some four non-intersecting sets of indices in $[9]$. Therefore, the code $\mathscr{C}$ is a $[9,4,4]_2$ batch code. As a matter of fact, this code is the two-layer construction of "subcube code" in [7, Section 3.2].

Lemma 1. *Let $\mathscr{C}$ be an $[M,n,m]_q$ batch code. Then, each row of G has Hamming weight at least m.*

Proof. Consider row j, for an arbitrary $j \in [n]$. We can retrieve the combination $(x_j, x_j, \cdots, x_j)$ if there are m non-intersecting sets of columns, such that sum of the symbols in each set is equal e_j^T. Therefore, there are at least m columns in G with a nonzero symbol in position j. □

Lemma 2. *Let $\mathscr{C}$ be an $[M,n,m]_q$ batch code. Then, the matrix G is full rank.*

Proof. We are able to recover any combination of size m of $\{x_1, x_2, \cdots, x_n\}$. Then, the column vectors

$$\begin{pmatrix}1\\0\\0\\\vdots\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\\vdots\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\\vdots\\0\end{pmatrix}, \quad \cdots \quad \begin{pmatrix}0\\0\\0\\\vdots\\1\end{pmatrix}$$

are all in the column space of G. Therefore, the column space of G has dimension n, and so the matrix is full rank. □

The following theorem is the main result of this section. The presented proof of this theorem works only for *binary* batch codes.

Theorem 2. *Let $\mathscr{C}$ be an $[M,n,m]_2$ batch code $\mathscr{C}$ over $\mathbb{F}_2$. Then, G is a generator matrix of the classical error-correcting $[M,n,\geq m]_2$ code.*

Proof. Let $\mathbb{C}$ be a classical ECC, whose generating matrix is G. It is obvious that the length of $\mathbb{C}$ is M. Moreover, since the matrix G is a full rank matrix due to Lemma 2, we obtain that the dimension of $\mathbb{C}$ is n. Thus, the only parameter in question is the minimum distance of $\mathbb{C}$.

In order to show that the minimum distance of $\mathbb{C}$ is at least m, it will be sufficient to show that any non-zero linear combination of the rows of G has Hamming weight at least m. Consider an arbitrary linear combination of the rows of G, whose indices are given by a set $T \neq \varnothing$,

$$z = \sum_{i \in T} G_i \ .$$

Take an arbitrary index $i_0 \in T$. Due to the properties of the batch codes we should be able to recover $(x_{i_0}, x_{i_0}, \cdots, x_{i_0})$ from y. Therefore, there exist m disjoint sets of indices $S_1, S_2, \cdots, S_m$, $S_i \subseteq [M]$, such that for all $i \in [m]$:

$$\sum_{j \in S_i} G^{[j]} = e_{i_0}^T \ . \tag{1}$$

Now, consider the sub-matrix M_i of G which is formed by the rows of G indexed by T and the columns of G indexed by S_i. Due to (1), the row of M_i that corresponds to the row i_0 in G, has an odd number of ones in it. All other rows of M_i contain an even number of ones. Therefore, the matrix M_i contains an odd number of ones. This means that the vector of z will also contain an odd number of ones in the positions given by the set S_i. This odd number is at least one.

We conclude that z contains at least one '1' in positions given by the set S_i, for all $i \in [m]$. The sets S_i are disjoint, and therefore the Hamming weight of z is at least m. □

Example 2. The converse of Theorem 2 is generally not true. In other words, if G is a generator matrix of a classical error-correcting $[M,n,m]_2$ code, then the corresponding code $\mathscr{C}$ is not necessarily an $[M,n,m]_2$ batch code. For example, take G to be a generator matrix of the classical $[4,3,2]_2$ ECC as follows:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} .$$

Let $x = (x_1, x_2, x_3)$, $y = (y_1, y_2, y_3, y_4) = xG$.

It is impossible to retrieve (x_2, x_3). This can be verified by the fact that

$$x_2 = y_1 + y_2 = y_3 + y_4 \quad \text{and} \quad x_3 = y_1 + y_3 = y_2 + y_4 ,$$

and so one of the y_i's is always needed to compute each of x_2 and x_3.

Corollary. The topic of linear ECCs was very intensively studied over the years. Various well-studied properties of linear ECCs, such as MacWilliams identities [11], apply also to linear batch codes due to Theorem 2 (for $t = 1$, $M = N$ and $q = 2$). A variety of bounds on the parameters of ECCs, such as sphere-packing bound, Plotkin bound, Griesmer bound, Elias-Bassalygo bound, McEliece-Rodemich-Rumsey-Welch bound [13] (see also [14, Chapter 4], [12]) apply to the parameters of linear binary $[M,n,m]$ batch codes.

4 Constructions of New Codes

In this section we present several simple methods to construct new linear batch codes from the existing ones.

Theorem 3. *Let $\mathscr{C}_1$ be an $[M_1,n,m_1]_q$ batch code and $\mathscr{C}_2$ be an $[M_2,n,m_2]_q$ batch code. Then, there exists an $[M_1+M_2,n,m_1+m_2]_q$ batch code.*

Proof. Let G_1 and G_2 be $n \times M_1$ and $n \times M_2$ generator matrices corresponding to $\mathscr{C}_1$ and $\mathscr{C}_2$, respectively. Consider the following $n \times (M_1 + M_2)$ matrix

$$\hat{G} = [\ G_1 \mid G_2\ ] .$$

This matrix corresponds to a batch code of length $M_1 + M_2$ with n variables. It is sufficient to show that any combination of $m_1 + m_2$ variables can be retrieved. By the assumption, the first (any) m_1 variables can be retrieved from the first M_1 coordinates of y and the last m_2 variables can be retrieved from the last M_2 coordinates of y. This completes the proof. □

Theorem 4. *Let $\mathscr{C}_1$ be an $[M_1, n_1, m_1]_q$ batch code and $\mathscr{C}_2$ be an $[M_2, n_2, m_2]_q$ batch code. Then, there exists an $[M_1 + M_2, n_1 + n_2, \min\{m_1, m_2\}]_q$ batch code.*

Proof. As before, denote by G_1 and G_2 the $n_1 \times M_1$ and $n_2 \times M_2$ generator matrices corresponding to $\mathscr{C}_1$ and $\mathscr{C}_2$, respectively. Consider the following $(n_1 + n_2) \times (M_1 + M_2)$ matrix

$$\hat{G} = \left[\begin{array}{c|c} G_1 & 0 \\ \hline 0 & G_2 \end{array}\right] .$$

The matrix $\hat{G}$ corresponds to a batch code of length $M_1 + M_2$ with $n_1 + n_2$ variables. Moreover, any combination of $\min\{m_1, m_2\}$ variables can be retrieved. If all unknowns are from $\{x_1, x_2, \cdots, x_{n_1}\}$, then they can be retrieved by using only the first M_1 columns of $\hat{G}$. If all unknowns are from $\{x_{n_1+1}, x_{n_1+2}, \cdots, x_{n_1+n_2}\}$, then they can be retrieved by using only the last M_2 columns of $\hat{G}$. Generally, some unknowns can be retrieved by using combinations of the first M_1 columns, while the other unknowns are retrieved using combinations of the last M_2 columns. Since the number of unknowns is at most $\min\{m_1, m_2\}$, we can always retrieve all of them simultaneously. □

Theorem 5. *Let $\mathscr{C}$ be an $[M, n, m]_q$ batch code, and let G be the corresponding $n \times M$ matrix. Then, the code $\hat{\mathscr{C}}$, defined by the $(n+1) \times (M+m)$ matrix*

$$\hat{G} = \underbrace{\left(\begin{array}{c|cccc} & 0 & 0 & \cdots & 0 \\ G & \vdots & \vdots & \ddots & \vdots \\ & 0 & 0 & \cdots & 0 \\ \hline \bullet\ \bullet\ \bullet\ \cdots\ \bullet & 1 & 1 & \cdots & 1 \end{array}\right)}_{M \qquad\qquad m}$$

is an $[M + m, n + 1, m]$ batch code, where $\bullet$ stands for an arbitrary symbol in $\mathbb{F}_q$.

Proof. As before, let $x = (x_1, x_2, \cdots, x_n, x_{n+1})$ and $y = (y_1, y_2, \cdots, y_{M+m}) = x\hat{G}$. Assume that we want to retrieve the vector $z = (x_{i_1}, x_{i_2}, \cdots, x_{i_m})$.

Take a particular x_{i_j} in z, $j \in [m]$. Consider two cases. If $i_j \neq n + 1$ then, since $\mathscr{C}$ is a batch code, we have

$$x_{i_j} = \sum_{\ell \in T_{i_j}} y_\ell + \xi \cdot x_{n+1} ,$$

where $T_{i_j} \subseteq [M]$ and $\xi \in \mathbb{F}_q$. In that case, if $\xi = 0$, then $x_{i_j} = \sum_{\ell \in T_{i_j}} y_\ell$. If $\xi \neq 0$, then $x_{i_j} = \sum_{\ell \in T_{i_j}} y_\ell + \xi \cdot y_{M+j}$. Observe that all T_{i_j} are disjoint due to the properties of a batch code.

In the second case, $i_j = n + 1$, and we simply set $x_{i_j} = x_{n+1} = y_{M+j}$.

In both cases, we used sets $\{y_\ell : \ell \in T_{i_j} \cup \{M + j\}\}$ to retrieve x_{i_j}. These sets are all disjoint for $j \in [m]$.

We conclude that all m unknowns x_{i_j}, $j \in [m]$, can be retrieved simultaneously. $\square$

Acknowledgements We thank Dominique Unruh for helpful discussions. The work of the authors is supported in part by the research grants PUT405 and IUT2-1 from the Estonian Research Council and by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science, EXCS. The work of V. Skachek is also supported in part by the EU COST Action IC1104.

References

1. Z. Bar-Yossef, Y. Birk, T.S. Jayram and T. Kol *Index coding with side information, IEEE Trans. Inform. Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011.
2. S. Bhattacharya, S. Ruj, and B. Roy, *Combinatorial batch codes: a lower bound and optimal constructions, Advances in Mathematics of Communications*, vol. 6, no. 2, pp. 165–174, 2012.
3. R.A. Brualdi, K. Kiernan, S.A. Meyer, and M.W. Schroeder, *Combinatorial batch codes and transversal matroids, Advances in Mathematics of Communications*, vol. 4, no. 3, pp. 419–431, 2010.
4. C. Bujtás and Z. Tuza, *Batch codes and their applications, Electronic Notes in Discrete Mathematics*, vol. 38, pp. 201–206, 2011.
5. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, *Private information retrieval, Proc. 36th Symp. on Foundations of Comp. Science (FOCS)*, pp. 41–50, 1995.
6. A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright, and K. Ramchandran *Network coding for distributed storage systems, IEEE Trans. Inform. Theory*, vol. 59, no. 9, pp. 4539–4551, Sept. 2010.
7. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, *Batch codes and their applications, Proc. of the 36th ACM Symposium on Theory of Computing (STOC)*, June 2004, Chicago, IL.
8. E. Kushilevitz and R. Ostrovsky, *Replication is NOT needed: SINGLE database, computationally-private information retrieval, Proc. 38th Symp. on Foundations of Comp. Science (FOCS)*, pp. 364–373, 1997.
9. H. Lipmaa, *First CPIR protocol with data-dependent computation, Proc. International Conference on Information Security and Cryptology (ICISC)*, pp. 193–210, 2009.
10. H. Lipmaa and V. Skachek, *Linear batch codes,* available online `http://arxiv.org/abs/1404.2796`.
11. F.J. MacWilliams, *A theorem on the distribution of weights in a systematic code, Bell System Tech. J.*, 42 (1963), 79–94.
12. F. J. MacWilliams and N. J. A. Sloane, *"The Theory of Error-Correcting Codes,"* Amsterdam, The Netherlands; North-Holland, 1978.
13. R.J. McEliece, E.R. Rodemich, H. Rumsey, and L.R. Welch *New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities, IEEE Trans. Inform. Theory*, vol. IT-23, pp. 157–166, Mar. 1997.
14. R. M. Roth, *"Introduction to Coding Theory,"* Cambridge University Press, Cambridge, United Kingdom, 2006.
15. N. Silberstein and A. Gál, *Optimal Combinatorial Batch Codes based on Block Designs,* available online `http://arxiv.org/abs/1312.5505`.
16. D. Stinson, R. Wei, and M. Paterson, *Combinatorial batch codes, Advances in Mathematics of Communications*, vol. 3, no. 1, pp. 13–17, 2009.