Chapter 13
Matrix Reduction

We have seen how mathematical reasoning can be used to compute persistent homology for simple filtrations. In this section, we present a more generally applicable computer algorithm that constructs persistence diagrams through matrix reduction. It can be viewed as a common extension of the incremental algorithm, discussed two sections ago, and the classic normal form algorithm, which can also be found in standard texts on algebraic topology [1].

13.1 Boundary Matrices

The input to the algorithm are the boundary maps represented by binary matrices. Recall that the p- and $(p-1)$-simplices form bases of the vector spaces of p- and $(p-1)$-chains. The p-th boundary map is fully specified by the p-th *boundary matrix*,

$$\partial_p = \partial_p[1 \ldots s_{p-1}, 1 \ldots s_p], \tag{13.1}$$

whose columns correspond to the p-simplices, whose rows correspond to the $(p-1)$-simplices, and whose entries record the face relation, with $\partial_p[i, j] = 1$ if the i-th $(p-1)$-simplex is a face of the j-th p-simplex, and $\partial_p[i, j] = 0$, otherwise. For example, the four boundary matrices of the tetrahedron are

$$\partial_0 = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}, \tag{13.2}$$

$$\partial_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}, \tag{13.3}$$

H. Edelsbrunner, *A Short Course in Computational Geometry and Topology*, SpringerBriefs in Mathematical Methods, DOI: 10.1007/978-3-319-05957-0_13,

$$\partial_2 = \begin{bmatrix} 0&0&1&1 \\ 0&1&0&1 \\ 1&0&1&0 \\ 1&1&0&0 \\ 1&0&0&1 \\ 0&1&1&0 \end{bmatrix}, \tag{13.4}$$

$$\partial_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{13.5}$$

We use standard row and column operations to reduce each matrix to *normal form*. By this we mean a matrix that is entirely zero, except for an initial segment of the diagonal, which consists of 1s. Since we use modulo 2 arithmetic, it is easy enough to get the matrices into this form. For the four boundary matrices above, we get

$$R_0 = \begin{bmatrix} 0&0&0&0 \end{bmatrix}. \tag{13.6}$$

$$R_1 = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}, \tag{13.7}$$

$$R_2 = \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix}, \tag{13.8}$$

$$R_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{13.9}$$

The dimensions of the null-spaces are the ranks of the cycle groups, the number of 1s in the diagonals are the ranks of the boundary groups in one lower dimension, and we get the Betti numbers by subtraction Table 13.1:

Table 13.1 The ranks of the cycle and boundary groups and the Betti numbers of the tetrahedron

	$p=0$	$p=1$	$p=2$	$p=3$
rank Z_p	4	3	1	0
rank B_p	3	3	1	0
β_p	1	0	0	0

13.2 Reduction Algorithm

To compute the Betti numbers, we need no particular ordering on the simplices, but for computing persistence, the ordering is essential. We therefore modify the classic algorithm in two ways, storing all boundary maps in one big matrix, and using only left-to-right column additions and no row operations. These are not enough to transform the matrix into normal form, but we will see that they suffice to give the needed rank information.

Let σ_1 to σ_n be the filter, and write $\partial = \partial[1 \dots n, 1 \dots n]$ for the big boundary matrix, with $\partial[i, j] = 1$ if $\dim \sigma_i = \dim \sigma_j - 1$ and σ_i is a face of σ_j, and $\partial[i, j] = 0$, otherwise. The algorithm reduces the matrix by iterating through the columns from left to right. It zeroes out a column j from bottom to top, to the extent possible. To explain this, we set $\text{pivot}(j)$ equal to the maximum row index for which column j has a non-zero entry. We set $\text{pivot}(j) = 0$ is the entire column is zero.

```
R = ∂;
for j = 1 to n do
  while ∃k < j with pivot(k) = pivot(j) ≠ 0 do
    add column k to column j
  endwhile
endfor.
```

We call the resulting matrix *reduced* because each row contains at most one pivot. We shortly discuss why the pivots are significant.

While the above pseudo-code states only two loops explicitly, there are really three. The innermost loop adds two columns of length n in time proportional to n. The middle loop searches for a column to the left of column j that has the pivot in the same row. It iterates until no such column can be found, and at each step, the pivot of column j moves up by at least one position. It follows that we iterate fewer than n times. Finally, the outer loop iterates over n columns. The total running time of the algorithm is therefore at most cubic in the number of simplices. All fast implementations of this algorithm use sparse matrix representations of ∂ and R. While ∂ is necessarily sparse, it is not clear that R inherits this property, but very often it does. Already the original publication on persistent homology [2] used a sparse matrix representation in which the columns are stored as lists that store only the non-zero entries. The availability of fast software is indeed an important factor in the success of the mathematical framework within applied communities.

13.3 Translation to Barcode

We illustrate the algorithm with an example, first turning the filtration in Fig. 11.1 into a single boundary matrix. We recall that we have a sequence of 15 simplices that correspond to the rows and the columns of the boundary matrix; see Fig. 13.1.

$$\partial = \begin{array}{c} \begin{array}{cccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \end{array} \\ \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & \boxed{1} & 0 & 0 & \boxed{1} & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & \boxed{1} & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & \boxed{1} & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & \boxed{1} & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & \boxed{1} & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & \boxed{1} & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 \end{array}\right] \end{array}$$

Fig. 13.1 The big boundary matrix of the filtration of the tetrahedron in Fig. 11.5

$$R = \begin{array}{c} \begin{array}{cccccccccccccccc} & & & & 4 & & & 4 & 4 & & & & & 12 & & \\ & & & & 5 & & & 5 & 5 & & 4 & & & 13 & & \\ & & & & 7 & & & 8 & 9 & & 7 & & 12 & 10 & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & \end{array} \\ \left[\begin{array}{cccccccccccccccc} 0 & 0 & 0 & \boxed{1} & \boxed{1} & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & \boxed{1} & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & \boxed{1} & \boxed{1} & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 \end{array}\right] \end{array}$$

Fig. 13.2 The reduced boundary matrix with lowest non-zero entries highlighted. The stacks of column indices at the *top* indicate the left-to-right column additions used to reduce the matrix

It should be clear that even just storing this matrix is a waste of resources because most entries are zero. However, the algorithm is easiest to explain for the full matrix, and we worry about speeding up the algorithm later.

The above matrix is small enough that we can run the algorithm by hand, displaying the result in Fig. 13.2. Each zero column is a cycle, adding to the rank of the corresponding cycle group. Together with the pivots, they encode all the information we need. The meaning of a pivot in row i and column $j+1$ is that the addition of σ_i creates a class that the addition of σ_{j+1} destroys. In other words, there is a homology class born at K_i that dies entering K_{j+1}. The dimension of the class is the dimension of σ_i. To translate this information into the barcode, we draw an interval from i to $j+1$ for every pivot in row i and column $j+1$, and we draw an interval from i to infinity if column i is zero but row i does not contain any pivot.

Translating the reduced matrix in Fig. 13.2 gives the barcode in Fig. 11.2. This is perhaps easiest to see by translating the reduced matrix into the persistence diagram in Fig. 11.3. To do this, we rotate the matrix by 45° in a ccw order so that its diagonal lies horizontal. The pivots become the points in the diagram, and we just need to add points at infinity for zero columns that do not correspond to pivots.

13.4 Uniqueness of Pivots

The number of pivots is the rank of the reduced matrix. Since column operations do not alter the rank, this number is also the rank of the initial boundary matrix. More than the number, we now show that the pivots themselves are unique and do not depend on the sequence of column operations used to reduce the matrix. To prove this, we consider lower-left submatrices ∂_i^j obtained from ∂ by deleting the first $i-1$ rows and the last $n-j$ columns. For example, $\partial = \partial_1^n$. Taking the alternating sum of the ranks of four submatrices gives

$$r_\partial(i, j+1) \quad = \quad \operatorname{rank} \partial_i^{j+1} + \operatorname{rank} \partial_{i+1}^{j} - \operatorname{rank} \partial_i^j - \operatorname{rank} \partial_{i+1}^{j+1}.$$

Substituting R for ∂, we get again submatrices and alternating sums of ranks. Since left-to-right column additions do not alter the ranks of any of these submatrices, $\operatorname{rank} \partial_i^j = \operatorname{rank} R_i^j$. Writing

$$\begin{aligned} A &= \operatorname{rank} R_i^{j+1}, & B &= \operatorname{rank} R_i^j, \\ C &= \operatorname{rank} R_{i+1}^{j+1}, & D &= \operatorname{rank} R_{i+1}^j, \end{aligned}$$

we therefore have $r_\partial(i, j+1) = r_R(i, j+1) = A - B + C - D$; see Fig. 13.3.

Pivot Uniqueness Lemma *An entry $R[i, j+1]$ is the pivot of column $j+1$ in the reduced boundary matrix iff $r_\partial(i, j+1) = 1$.*

Proof Since $r_\partial = r_R$, it suffices to show $\operatorname{pivot}(j+1) = i$ iff $r_R(i, j+1) = 1$. We prove this in a case analysis, making use of the fact that the non-zero columns in R and any of its lower-left submatrices are linearly independent.

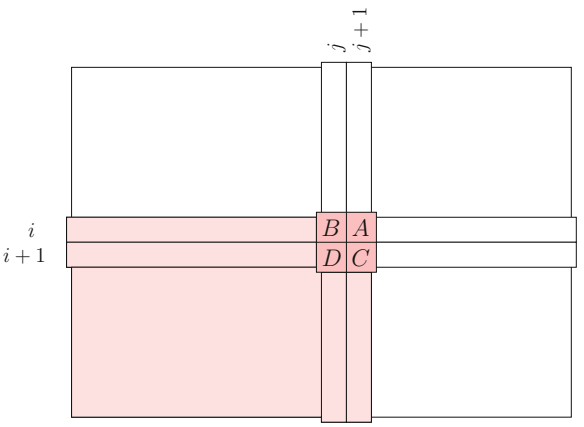

Fig. 13.3 Four submatrices of ∂ or of R, and their ranks

First, assume $\text{pivot}(j+1) = i$. Since the last column is non-zero, we have $A - B = 1$. If we now delete the top row from R_i^{j+1}, then the last column is zero, implying $C - D = 0$. Hence, $r_R(i, j+1) = 1$, as required. Second, assume $\text{pivot}(j+1) \neq i$. If $\text{pivot}(j+1) < i$, then the last column is zero, and we get $A - B = C - D = 0$. Else if $\text{pivot}(j+1) > i$, then the last column is non-zero, even after deleting the top row, and we again get $A - B = C - D = 0$. The claimed result follows. □

An off-shot of this analysis is an explanation why the reduction algorithm does not continue the elimination of non-zero entries above the pivots.

13.5 Alternative Algorithm

The Pivot Uniqueness Lemma can also be used to justify variants of the algorithm. For example, we could use the current column to reduce future columns, as opposed to using past columns to reduce the current column.

```
R = ∂;
for j = 1 to n do
  i = pivot(j);
  for ℓ = j + 1 to n do
    if R[i, ℓ] = 1 then add column j to column ℓ endif
  endfor
endfor.
```

This works provided we stipulate that $R[0, \ell] = 0$ for all columns ℓ. The running time is similar to the earlier algorithm, namely at most cubic in the number of simplices. While the resulting reduced matrices computed by the two algorithms

may be different, the Pivot Uniqueness Lemma implies that the sets of pivots are the same. This is comforting because it confirms that the persistence diagram is uniquely defined by the filtration and does not depend on the algorithm that computes it.

References

1. Munkres JR (1984) Elements of algebraic topology. Perseus Publications, Cambridge, Massachusetts
2. Edelsbrunner H, Letscher D, Zomorodian A (2002) Topological persistence and simplification. Discrete Comput Geom 28:511–533