1 Solving parameters using Gauss–Jordan elimination

Some approaches to solving determined systems of linear equations such as Cramer's rule, require enough constrains to ensure uniqueness of the solution, without it being overdetermined at the same time. Numerical methods may be difficult to control, and could add additional approximations to the solution. In our case, the set of available equations is greater than required number of equations that satisfy above conditions. Out of 2^n possible binary vectors representing conditional probability of events, we are forced to pick n that introduce the smallest error in further calculations. This approach can yield good results when done properly, but the question of which equations to choose remains unanswered. Preserving linear independence of vectors invokes additional complexity to the rules by which we choose the final set of equations.
Using Gauss–Jordan elimination, we can avoid this problem entirely, since it allows us to work with both overdetermined and underdetermined systems of equations. The order of equations is also taken into account, so in cases of contradictions, certain combinations are preferred to others. For the remaining part of this section we will work with linear equations, since using product equations would require us to redefine elementary row operations, thus introducing unnecessary complications.

1.1 Example of Gauss–Jordan elimination

Let us propose a simple equation set presented in a standard matrix form $A \cdot X = b$.

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \tag{1}$$

We will use symbols for vector b, so we can keep track of operations done on absolute terms. Turning that into augmented matrix $[A|b]$ yields

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] \tag{2}$$

Notice that this equation set may be contradictory if we were to use real numbers – x_3 can be calculated as a linear combination of two first rows ($x_3 = b_1 - b_2$) or by taking fourth row as a solution ($x_3 = b_4$). It is plausible that substituting b_1, b_2 and b_3 with real values calculated from the data file or a database, would lead to x_3 being equal to two different numbers. Additionally, the equation set is underdetermined – there is not enough information about x_1 and x_2 to solve them independently.
We will now perform Gauss–Jordan elimination steps in order to show that certain properties we care about (such as preserving preference of equations determined by their order) are taken into account.
We will distinguish pivot elements with colors red (currently selected pivot element) and blue (previous pivot elements). **During the algorithm we will prefer the topmost pivot element in given column – this way we assure that topmost equations will be more significant**

1. We choose the first pivot element (in red), and use it to zero-out remaining coefficients in first column.

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] r_2 = r_2 - r_1 \sim \left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 0 & 0 & -1 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] \tag{3}$$

2. We select the second pivot element - note that no two pivot elements can share the same row. The first non-zero element that satisfies this condition is coefficient of x_3 in the second row.

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 0 & 0 & -1 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] r_2 = r_2 \cdot (-1) \sim \left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] \tag{4}$$

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] \begin{array}{l} r_1 = r_1 - r_2 \\ \\ \\ r_4 = r_4 - r_2 \end{array} \sim \left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & b_1 - (b_1 - b_2) \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 0 & b_4 - (b_1 - b_2) \end{array}\right] \tag{5}$$

3. The last pivot element is going to be coefficient at x_4 in the third row. Since the remaining coefficients are all zeros in fourth column, no changes are made. We can simplify the values in new vector b. Notice that fourth row is a zero-vector - we can eliminate that from the equation set.

$$\left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & b_1 - (b_1 - b_2) \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 0 & b_4 - (b_1 - b_2) \end{array}\right] \sim \left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \end{array}\right] \tag{6}$$

Let us compare our end-result with the initial matrix:

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] \tag{7}$$

As we can see, x_3 was calculated using first and the second row $(b_1 - b_2)$. Let us see what happens after we move the third row on the top position, indicating that our preferred ordering of equations changed. (We expect now to calculate x_3 solely by first row).

1. We choose the our first pivot element, and use it to zero-out remaining coefficients in first column.

$$\left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 1 & 0 & b_2 \\ 1 & 1 & 0 & 0 & b_3 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] r_3 = r_3 - r_2 \sim \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 1 & 0 & b_2 \\ 0 & 0 & -1 & 0 & b_3 - b_2 \\ 0 & 0 & 0 & 1 & b_1 \end{array}\right] \tag{8}$$

2. Again, no candidate for pivot element in the second column, coefficient at x_3 in first row is the next pivot element

$$\left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 1 & 0 & b_2 \\ 0 & 0 & -1 & 0 & b_3 - b_2 \\ 0 & 0 & 0 & 1 & b_1 \end{array}\right] \begin{array}{l} r_2 = r_2 - r_1 \\ r_3 = r_3 + r_1 \end{array} \sim \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 0 & b_3 - b_2 + b_1 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] \tag{9}$$

3. Last item fit for a pivot element is a coefficient at x_4 in fourth row. After getting rid of zero vectors, we achieve the following reduced row echelon form matrix

$$\left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 0 & b_3 - b_2 + b_1 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] \sim \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] \tag{10}$$

As expected, x_3 was calculated using the most preferred set of equations, as dictated by their order. What this method does not take into account is the relative weight of each row. As of yet, all we could rely on was simple ordering of equations, without using the information about quantity or frequency of each type of equation in our learning set. If our method was to provide that feature to us as well, we could talk about very complete and solid solution that can be expected to perform optimally.

1.2 Properties of zero vectors

In this section we will describe how solving the equation set using one order does not prevent us from reproducing other solutions for given parameter. As we saw previously, different order of equations may lead to different outcomes for certain parameters. This variety came from contradictions in the equation set.

Zero vector

We will use the term "zero vector" to describe the vectors of the form

$$\left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & M(b) \end{array}\right], \tag{11}$$

where $M(b)$ is a linear combination of absolute terms (vector b).

Zero vectors that emerge during Gauss–Jordan elimination contain information about other possible solutions for given value. Instead of removing them in the process, we can store them, and utilize them later. Let us see how previous example holds to that theory.

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \end{array}\right] \sim \left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & b_1 - (b_1 - b_2) \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \\ 0 & 0 & 0 & 0 & b_4 - (b_1 - b_2) \end{array}\right] \sim \left[\begin{array}{cccc|c} 1 & 1 & 0 & 0 & b_2 \\ 0 & 0 & 1 & 0 & b_1 - b_2 \\ 0 & 0 & 0 & 1 & b_3 \end{array}\right] \tag{12}$$

This particular order of equation lead to x_3 being calculated from two top-most equations in a set. If we add our final solution for x_3 (vector in red), to the zero vector we ought to remove in a penultimate step of our algorithm (vector in blue), we obtain previously abandoned solution:

$$\left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 - b_2 \end{array}\right] + \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & b_4 - (b_1 - b_2) \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_4 \end{array}\right] \tag{13}$$

Using the same method we can start from the solution obtained after rearranging the order of equations in the initial matrix.

$$\left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 1 & 0 & b_2 \\ 1 & 1 & 0 & 0 & b_3 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] \sim \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 0 & b_3 - b_2 + b_1 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] \sim \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \\ 1 & 1 & 0 & 0 & b_2 - b_1 \\ 0 & 0 & 0 & 1 & b_4 \end{array}\right] \tag{14}$$

Linear combination of two vectors again yields a different solution

$$\left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_1 \end{array}\right] + (-1) \cdot \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & b_3 - b_2 + b_1 \end{array}\right] = \left[\begin{array}{cccc|c} 0 & 0 & 1 & 0 & b_2 - b_3 \end{array}\right] \tag{15}$$

Let us look at the example of the equation set with multiple zero vectors:

$$\left[\begin{array}{cccc|c} 1 & 1 & 0 & 1 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 1 & 1 & 0 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \\ 1 & 0 & 0 & 0 & b_5 \\ 1 & 0 & 0 & 1 & b_6 \end{array}\right] \tag{16}$$

This equation set is overconstrained – we can expect to obtain at least two zero vectors after the Gauss–Jordan elimination steps.

$$\left[\begin{array}{cccc|c} 1 & 1 & 0 & 1 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 1 & 1 & 0 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \\ 1 & 0 & 0 & 0 & b_5 \\ 1 & 0 & 0 & 1 & b_6 \end{array}\right] \sim \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & b_2 - b_3 + b_4 \\ 0 & 0 & 0 & 1 & b_1 - b_2 \\ 0 & 1 & 0 & 0 & b_3 - b_4 \\ 0 & 0 & 1 & 0 & b_4 \\ 0 & 0 & 0 & 0 & -b_2 + b_3 - b_4 + b_5 \\ 0 & 0 & 0 & 0 & -b_1 + b_3 - b_4 + b_6 \end{array}\right] \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \tag{17}$$

Let us mark each vector in a reduced row echelon form (Equation 17) as $v_1 \dots v_6$. As we had shown previously, we can use zero vectors to obtain different solutions to parameters $x_1 \dots x_4$, for example:

	combination	value
$[1 \quad 0 \quad 0 \quad 0]_1$	v_1	$b_2 - b_3 + b_4$
$[1 \quad 0 \quad 0 \quad 0]_2$	$v_1 + v_5$	b_5
$[1 \quad 0 \quad 0 \quad 0]_3$	$v_1 + v_6$	$-b_1 + b_2 + b_6$
$[0 \quad 1 \quad 0 \quad 0]_1$	v_3	$b_3 - b_4$
$[0 \quad 1 \quad 0 \quad 0]_2$	$v_3 - v_5$	$b_2 - b_5$
$[0 \quad 1 \quad 0 \quad 0]_3$	$v_3 - v_6$	$b_1 - b_6$
$\dots$	$\dots$	$\dots$
$[0 \quad 0 \quad 0 \quad 1]_1$	v_2	$b_1 - b_2$
$[0 \quad 0 \quad 0 \quad 1]_2$	$v_2 + v_6 - v_5$	$b_6 - b_5$
$\dots$	$\dots$	$\dots$

(18)

Notice that the second solution for x_4 (in red above) requires two zero vectors to find an efficient solution.

We would like to propose a conjecture describing relationship of possible solutions with zero vectors in reduced row echelon form matrix.

Zero-vector conjecture

Every possible solution for given parameter q can be obtained as a linear combination of a solution vector from Gauss–Jordan elimination, and the zero vectors, i.e.,

$$\begin{gathered}\forall_{s\in S}\exists_{a\in V}\ s = [1, a_1, a_2, \ldots, a_n] \cdot [s_0, z_1, z_2, \ldots, z_n] = \\ = s_0 + a_1 \cdot z_1 + a_2 \cdot z_2 + \cdots + a_n \cdot z_n\end{gathered} \tag{19}$$

where $s \in S$ is a solution vector s for parameter q from a space of all possible solutions S
$a \in V$ is the vector of coefficients from a vector space V over a field $\mathbb{R}$
s_0 is the first solution (also a vector over a field $\mathbb{R}$) for given parameter, as obtained from Gauss–Jordan elimination
z_i is the i-th zero vector obtained from Gauss–Jordan elimination ($i \in 1 \ldots n$).
n is the number of zero vectors in reduced row echelon form.

Proof

If the equation set is determined, each parameter has a unique solution. Since in that case there are no zero vectors, $n = 0 => s = [1] \cdot [s_0] = s_0$.
Similar case would emerge when the equation set is strictly underdetermined (not every parameter has a unique solution, but no zero vectors appear in reduced row echelon form either).
Third case would be equation sets with over-constrainments, which are of our interest here since they produce zero vectors after Gauss–Jordan elimination.

First, let's define two terms we will later use:

Linear combination of equation set

Linear combination of the equation set can be interpreted as a function

$$f : \mathbb{M}_{m\times n} \to \mathbb{M}_{m\times n} \tag{20}$$

where $\mathbb{M}_{m\times n}$ is a space of matrices of size $m \times n$.

Additionally every such function f is equivalent to left multiplication by some matrix F, that is

$$\forall_f \forall_{A_0\in\mathbb{M}_{m\times n}} \exists_{F\in\mathbb{M}_{m\times m}}\ f(A_0) = F \cdot A_0 \tag{21}$$

Example: Equivalence relation 17 could also be written as an equation:

$$\begin{bmatrix} 0 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 1 & 0 \\ -1 & 0 & 1 & -1 & 0 & 1 \end{bmatrix} \cdot \left[\begin{array}{cccc|c} 1 & 1 & 0 & 1 & b_1 \\ 1 & 1 & 0 & 0 & b_2 \\ 0 & 1 & 1 & 0 & b_3 \\ 0 & 0 & 1 & 0 & b_4 \\ 1 & 0 & 0 & 0 & b_5 \\ 1 & 0 & 0 & 1 & b_6 \end{array}\right] = \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & b_2 - b_3 + b_4 \\ 0 & 0 & 0 & 1 & b_1 - b_2 \\ 0 & 1 & 0 & 0 & b_3 - b_4 \\ 0 & 0 & 1 & 0 & b_4 \\ 0 & 0 & 0 & 0 & -b_2 + b_3 - b_4 + b_5 \\ 0 & 0 & 0 & 0 & -b_1 + b_3 - b_4 + b_6 \end{array}\right] \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} \tag{22}$$

in which case the leftmost matrix would be our linear combination of Gauss–Jordan elimination algorithm.

Solution vector

Solution vector is a single row in a matrix (usually obtained by linear combination of the equation set), directly solving given parameter x_k (vector $[\ 0\ \ldots\ 0\ 1\ 0\ \ldots 0 \mid b\ ]$ with "1" at the k-th place, and some absolute term b in its augmented form).

Example: Row v_2 in equation 22 unambiguously gives solution to parameter x_4, thus vector $[\ 0\ 0\ 0\ 1 \mid b_1 - b_2\ ]$ is the solution vector of x_4.

We will now prove the conjecture in question.

Let $\boldsymbol{A}$ be the original equation set, and $\boldsymbol{B}$ - the equation set after Gauss–Jordan elimination (reduced row echelon form). Let us say that given parameter x_m can be calculated using at least two different linear combinations of vectors from $\boldsymbol{A}$. Let us call these L_0 and L_k, where L_0 is linear combination equivalent to Gauss–Jordan elimination ($\boldsymbol{B} = L_0 \cdot \boldsymbol{A}$). Let us assume that s_0 is the solution vector for parameter x_m as obtained from Gauss–Jordan elimination (that is $s_0 \in \boldsymbol{B}$)

s_0 is a solution vector in in $\boldsymbol{B}$.

s_k is a solution vector in $L_k \cdot \boldsymbol{A}$, but it is not a vector in $\boldsymbol{B}$.

We can show that s_k can also be obtained as linear combination of vectors from $\boldsymbol{B}$ using only s_0 and the zero vectors, by splitting the conjecture into two parts:

1. Vector s_k can be obtained as a linear combination of vectors from $\boldsymbol{B}$:
   Since L_0 is determined by Gauss–Jordan elimination, which in turn uses only elementary row operations, L_0 is an invertible matrix. Thus L_0^{-1} exists. Because $L_0^{-1} \cdot L_0 = \mathrm{Id}$, and matrix multiplication is associative, we can apply the following:

   $$s_k \in L_k \cdot \boldsymbol{A} \Rightarrow s_k \in L_k \cdot (L_0^{-1} \cdot L_0) \cdot \boldsymbol{A} \Rightarrow s_k \in L_k \cdot L_0^{-1} \cdot \boldsymbol{B} \tag{23}$$

   Because s_k is a vector in $L_k \cdot \boldsymbol{A}$ then s_k is also a vector in $L_k \cdot L_0^{-1} \cdot \boldsymbol{B}$. In that case we can use a linear combination $L_k \cdot L_0^{-1}$ to go from solution s_0 to s_k.
2. Vector s_k in $L_k \cdot L_0^{-1} \cdot \boldsymbol{B}$ is a linear combination of vectors no other than s_0 and the zero vectors in $\boldsymbol{B}$:
   (Quite hand-wavy argument I'm not satisfied with yet)

   Because $\boldsymbol{B}$ is a reduced row echelon form, the following property holds: no two non-zero coefficients in reduced row echelon form share the same column.

   Let us assume that vector s_k is calculated using two non-zero vectors in its linear combination from $\boldsymbol{B}$ to $L_k \cdot \boldsymbol{A}$. In that case, s_0 has to appear in a linear combination with a non-zero coefficient since no other non-zero vector can produce a 1 in m-th column in s_k. Additionally, any linear combination involving any two non-zero vectors from $\boldsymbol{B}$ with both coefficients other than 0 will produce a vector with at least two coefficients other than 0. Such vector would not be a solution vector, which s_k is. Contradiction.

Finding such linear combination is no trivial task. The solution space is infinite, and virtually any linear combination of zero vectors can be added to any non-zero vector, giving us an valid solution (although the combination would be very inefficient in most cases)

1.3 Modifying systems of equations

In previous sections we showed that we can use Gauss-Jordan elimination to explore space of solution for each parameter in fairly controlled environment. We have to remember however that our equations are not standard linear equations, but product equations. By product equations we mean equations such as

$$1 - (1 - p_1) \cdot (1 - p_2) \cdots (1 - p_n) = b_k. \tag{24}$$

In order to simplify the equation, we introduce substitutions $\forall_k q_k = 1 - p_k$ and $c_k = 1 - b_k$. This gives us equation of form

$$q_1 \cdot q_2 \cdots q_n = c_k. \tag{25}$$

Of course not every cause is present in given case, resulting in some parameters not taking part in the product on the left side of the equation. Let us propose an example of such equation:

$$\begin{cases} 1 - (1 - p_1) \cdot (1 - p_2) \cdot (1 - p_3) \cdot (1 - p_4) & = b_1 \\ 1 - (1 - p_3) \cdot (1 - p_4) & = b_2 \\ 1 - (1 - p_1) \cdot (1 - p_2) \cdot (1 - p_4) & = b_3 \\ 1 - (1 - p_3) & = b_4 \end{cases} \sim \begin{cases} q_1 \cdot q_2 \cdot q_3 \cdot q_4 & = c_1 \\ q_3 \cdot q_4 & = c_2 \\ q_1 \cdot q_2 \cdot q_4 & = c_3 \\ q_3 & = c_4 \end{cases}. \tag{26}$$

Taking logarithm of both sides yields the following system:

$$\begin{cases} log_p q_1 + log_p q_2 + log_p q_3 + log_p q_4 & = log_p c_1 \\ log_p q_3 + log_p q_4 & = log_p c_2 \\ log_p q_1 + log_p q_2 + log_p q_4 & = log_p c_3 \\ log_p q_3 & = log_p c_4 \end{cases}. \tag{27}$$

Since this gives us a linear equation set, Gauss–Jordan elimination is applicable.

1.4 Eliciting leak parameter

In this section we will discuss our approach to eliciting leak parameter from the data file. Since leak is slightly different from ordinary parameters, we have to approach this endeavour with high cautious. Leak, when not expressed explicitly in the data file, is represented in the probability of all parent nodes being in the distinguished state with child node being activated.

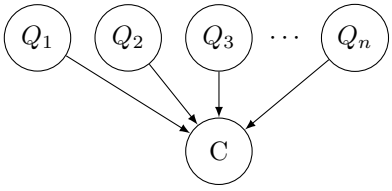

Figure 1.a: Structure of simple Bayesian Network

	Q_1	Q_2	Q_3	$\ldots$	Q_n
$P(Q_k = +)$	p_1	p_2	p_3	$\ldots$	p_n
$P(Q_k = -)$	$1 - p_1$	$1 - p_2$	$1 - p_3$	$\ldots$	$1 - p_n$

Figure 1.b: Prior probability of parent nodes

Let us assume that prior probability of parents $Q_1, \cdots, Q_n$ (see Figure 1.a) being in the distinguished stated is $(1-p_1), \cdots, (1-p_n)$ respectively. In that case, incidence of vector describing leak within the data file is equal to:

$$\prod_{i=1}^{n} (1 - p_i) \tag{28}$$

This is a trivial remark, since similar product can be derived for any combination of parent states. As we had shown previously, in case of other parameters we can aid our efforts of solving parameters by means of linear combination of vectors and linear algebra. We would like to achieve similar freedom in solving leak as well. This is especially important if parameters $p_1 \cdots p_n$ would be relatively high, favoring non-distinguished states to occur more often. Since in such cases records describing leak can be relatively rare, we can try to express leak as one of the explicit parameters. For this we will compare two approaches to interpreting leak within the data. (reference to canonical.pdf - diez and henrion approach to leak)

$$P(y|x) = 1 - (1 - p_L) \cdot \prod_{i \in I(x)} (1 - p_i) \tag{29}$$

$$P(y|x) = 1 - (1 - p_L) \cdot \prod_{i \in I(x)} \frac{1 - p'_i}{1 - p_L} \tag{30}$$

Equation 29 is the definition of leak by Diez[reference], while Equation 30 is the definition proposed by Henrion[reference]. We can make a transition between p_k and p'_k for any k using Equation 31.

$$1 - p_k = \frac{1 - p'_k}{1 - p_L} \tag{31}$$

Since main concern of our research is learning from data, Henrion's definition is to be applied, but because it is easy to transition between these, we can explore both ways and choose the one that yields better flexibility. Let us see how we would approach adding leak to the data:

#	P_1	P_2	P_3	P_4	Leak–Diez	Leak–Henrion	$P(y\|x)$
1	1	1	0	0	1	-1	p_1
2	0	1	1	1	1	-2	p_2
3	1	1	1	0	1	-2	p_3
4	0	0	1	0	1	0	p_4
5	0	0	0	0	1	1	p_5

Table 1: Example of data supplemented with explicit leak parameters using both approaches.

Interpreting leak by Diez's definition results in adding a column of ones to the data, indicating that leak was present in every case (this also derives from Equation 29).
The values in the column "Leak–Henrion" may be interpreted as exponents of each item in the product. Negative value $-k$ indicates positive exponent k in the denominator of the product (Equation 30). This rule still holds after we take logarithm of both sides, since then we treat the values in the data as coefficients of logarithms.

1. **Diez's definition:**
   Algorithm would be performed in three steps:
   (a) Solve the equation set for leak first
   (b) Apply transition 31 to the original parameters

(c) Solve the original equation (without leak) for each of the parameters

This would yield us the final parameters with leak included, as defined by Henrion.

2. **Henrion's definition:**
   Since the Equation 30 already suits our case of learning from data, solving the equation yields us the final parameters.

It may seem that second approach is simpler and handles the issues with leak for us, since there's no additional step of including leak to initial parameters from Diez's approach. Yet, since they are equivalent, there's no other way than to agree that potential error in leak will carry on to parameters with both approaches anyway. In that case, using Henrion's approach to supplement the data with leak is slightly disadvantageous, since it limits our solution for leak to the first solution obtained by Gauss–Jordan elimination.

1.5 Modified Gauss-Jordan elimination

In this subsection we will describe a slightly modified Gauss–Jordan elimination.
We can introduce a matrix notation for given product–equation set, with exponents of parameters as values:

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & c_1 \\ 0 & 0 & 1 & 1 & c_2 \\ 1 & 1 & 0 & 1 & c_3 \\ 0 & 0 & 1 & 0 & c_4 \end{array}\right] \tag{32}$$

Elementary row operation in standard Gauss–Jordan algorithm consist of

1. Row switching
2. Row addition (denoted as $\oplus$)
3. Row multiplication by a non-zero constant (denoted as $\odot$)

Row switching does not change in our modified Gauss-Jordan algorithm.
Row addition in regular Gauss–Jordan elimination is equivalent to adding coefficients of corresponding parameters. In order to preserve the same result in matrix notation (which in our case stores information about the exponents rather than coefficients), we multiply both sides of equations:

$$\left[\begin{array}{ccc|c} 1 & 0 & 2 & b_1 \\ -1 & 2 & 0 & b_2 \end{array}\right] r_1 = r_1 \oplus r_2 \quad \sim \left[\begin{array}{ccc|c} 0 & 2 & 2 & b_1 \oplus b_2 \\ -1 & 2 & 0 & b_2 \end{array}\right] \tag{33}$$

$$\left\{\begin{array}{ll} q_1 \cdot q_3^2 & = b_1 \\ q_1^{-1} \cdot q_2^2 & = b_2 \end{array}\right. \Bigg| \; r1 = r1 \cdot r2 \sim \left\{\begin{array}{ll} q_2^2 \cdot q_3^2 & = b_1 \cdot b_2 \\ q_1^{-1} \cdot q_2^2 & = b_2 \end{array}\right. \tag{34}$$

Row multiplication by a non-zero constant was previously equivalent to multiplication of each coefficient by a constant d. In order to multiply each exponent of each parameter, we simply raise both sides of equation to the power d.

$$\left[\begin{array}{ccc|c} 1 & 0 & 2 & b_1 \\ -1 & 2 & 0 & b_2 \end{array}\right] \begin{array}{c} r_1 = r_1 \odot \frac{1}{3} \\ r_2 = r_2 \odot (-3) \end{array} \sim \left[\begin{array}{ccc|c} \frac{1}{3} & 0 & \frac{2}{3} & b_1 \odot \frac{1}{3} \\ 3 & -6 & 0 & b_2 \odot (-3) \end{array}\right] \tag{35}$$

$$\begin{cases} q_1 \cdot q_3^2 & = b_1 \\ q_1^{-1} \cdot q_2^2 & = b_2 \end{cases} \left| \begin{array}{l} r_1 = r_1^{\frac{1}{3}} \\ r_2 = r_2^{-3} \end{array} \right. \sim \begin{cases} q_1^{\frac{1}{3}} \cdot q_3^{\frac{2}{3}} & = b_1^{\frac{1}{3}} \\ q_1^3 \cdot q_2^{-6} & = b_2^{-3} \end{cases} \tag{36}$$

This way we can define a new set of elementary operations as described in Table 2. The algorithm

Gauss–Jordan elimination	Modified Gauss–Jordan elimination
Row switching	Row switching
Row addition	Row multiplication
Row multiplication by a non-zero constant	Row exponentiation by a non-zero constant

Table 2: Elementary operations of modified Gauss–Jordan elimination

itself does not change as long as we apply the newly derived elementary row operations for exponents. Method proposed above is equivalent to standard Gauss–Jordan elimination, but does not require taking logarithm of both sides of equation. For simplicity, we will regard previous approach as the default one, but it was worth mentioning this approach as well for cases where it would simplify the implementation efforts.

2 Different approaches to solving parameters

Since our method enables us to propose different solutions for each parameter (in cases of overdetermined systems), we can propose few means of choosing the final value for given parameter. These can rely on choosing the best solution out of all that are possible, or combining several best choices together.

1. **Choose the best candidate for a solution out of k best guesses.**

   We have to remember that in cases of ambiguity for each parameter, there are infinitely many linear combinations of first solution and the zero vectors. If we would like to pick the best solution, we have to restrict our search space to the subset of k best candidates. In order to compare candidates for a solution, we can propose a fitness function that tries to estimate the error associated with each solution.

2. **Combine k possible candidates for solution into a final answer**

   Once we elicit the set of k possible candidates, we can use weighted average to combine them into final result. This will not automatically improve the result, but will average-out the error, which in most cases may result good solutions. This may be especially useful in case where it's difficult to propose a fitness function that describes the error accurately.

Since both approaches rely on similar input (fitness function, initial set of k candidates) we'll try to describe them first.

Candidates for the solution As we had shown previously, outcome of Gauss–Jordan elimination is just one of the possible solutions in overdetermined systems of equations.

Fitness function

1. **Relative frequency of each combination in a data file** Intuitively it is clear that the error is inversely proportional to the frequency of given combination within the data file. First good guess for a fitness function could be the frequency with which given equation appears in a data file.
2. **Fitness based on confidence interval** We define confidence interval for a fraction as follows:
$$P\left(\frac{x}{n} - u_\alpha\sqrt{\frac{\frac{x}{n}(1-\frac{x}{n})}{n}} < p < \frac{x}{n} + u_\alpha\sqrt{\frac{\frac{x}{n}(1-\frac{x}{n})}{n}}\right) = 1-\alpha, \tag{37}$$
where
   - x - size of population where $P(y = True|x)$ for given event x
   - n - size of the sample
   - u_α - coefficient derived from cumulative distribution function. (FIXME: severly requires a better translation)

   Since we require a fitness function, we can adapt the part responsible for the width of the interval as our fitness:
$$F'(x,n) = u_\alpha\sqrt{\frac{\frac{x}{n}(1-\frac{x}{n})}{n}} \tag{38}$$
   The solution is a linear combination of such fractions. We can take the maximum spear as our final fitness.
$$F(s) = \max\{F'(x,n)| \text{ for each } (x,n) \text{in solution}\} \tag{39}$$

2.1 Examples

Let us append some quality measures to each equation in the dataset. The whole equation set adds up to 2500 records. We can calculate a frequency for every vector, and treat it as a weight.

				probability	quantity	frequency Fq(b_i)
0	0	1	0	b_1	980	0.392
1	1	1	0	b_2	760	0.304
1	1	0	0	b_3	440	0.176
0	0	0	1	b_4	320	0.128
					2500	

(40)

Using this data we can propose few heuristics for calculating final value of x_3 or compare different solutions. Let us propose a fitness function for a solution:

$$F(s) = \frac{\prod_{b_i \in s} Fq(b_i)}{\sum_{b_i \in s} 1}, \tag{41}$$

, where $b_i \in s$ is true when b_i is taken into account (adding or subtracting) in given solution.

				probability	fitness function F(s)
$[0$	0	1	$0]_1$	b_1	0.392
$[0$	0	1	$0]_2$	$b_2 - b_3$	$\frac{0.304 \cdot 0.176}{2} = 0.24$

(42)

At this point we can pick a solution with a higher fitness value, or take weighted average of each solution as our final answer:

	probability
$[0 \quad 0 \quad 1 \quad 0]$	$\frac{0.392 \cdot b_1 + 0.24 \cdot (b_2 - b_3)}{0.392 + 0.24}$

(43)