Gaussian Elimination problems

Coding the Matrix, 2015

For auto-graded problems, edit the file Gaussian_Elimination_problems.py to include your solution.

Problem 1: Each of the matrices given below is almost in echelon form; give the matrix resulting from replacing the *MINIMUM* number of elements with zeroes to obtain a matrix in echelon form. You are not allowed to reorder the rows or columns of the matrix. (Note: you don't need to actually do any steps of Gaussian elimination for this problem.)

Example: Given the matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 9 & 2 & 3 & 4 \\ 0 & 0 & 3 & 4 \\ 0 & 8 & 0 & 4 \end{bmatrix}$$

you would replace the 9 and 8 with zeroes, so you answer would be the matrix

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

Okay, here are the problems:

1. $\begin{bmatrix} 1 & 2 & 0 & 2 & 0 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 2 & 3 & 4 \\ 1 & 0 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 & 4 \end{bmatrix}$

2. $\begin{bmatrix} 0 & 4 & 3 & 4 & 4 \\ 6 & 5 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$

3. $\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

4. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Problem 2: Write a procedure `is_echelon(A)` that takes a matrix in list of row lists and returns `True` if it is echelon form, and `False` otherwise.

Try out your procedure on the following matrices:

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{(True)}, \quad \begin{bmatrix} 2 & 1 & 0 \\ -4 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{(False)}, \quad \begin{bmatrix} 2 & 1 & 0 \\ 0 & 3 & 0 \\ 1 & 0 & 1 \end{bmatrix} \text{(False)}, \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 1 & 3 \\ 0 & 0 & 0 & 5 & 3 \end{bmatrix} \text{(True)}$$

Solving Echelon-Matrix-Vector Equations

Consider solving the matrix-vector equation

$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline 0 & 1 & & 1 & & \\ 1 & & 2 & & 3 & \\ 2 & & & & 1 & 9 \end{array} * [x_a, x_b, x_c, x_d, x_e] = [1, 1, 1]$$

where the matrix is in echelon form. The algorithm for solving this is very similar to the algorithm for solving a triangular system. The difference is that the algorithm disregards some of the columns. In particular, any column not containing the leftmost nonzero entry of some row is disregarded.

For the above example, the algorithm should disregard columns c and e. Column a contains the leftmost nonzero in row 0, column b contains the leftmost nonzero in row 1, and column d contains the leftmost nonzero in row 2.

Problem 3: For each of the following matrix-vector equations, find the solution:

(a) $\begin{bmatrix} 10 & 2 & -3 & 53 \\ 0 & 0 & 1 & 2013 \end{bmatrix} * [x_1, x_2, x_3, x_4] = [1, 3]$

(b) $\begin{bmatrix} 2 & 0 & 1 & 3 \\ 0 & 0 & 5 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} * [x_1, x_2, x_3, x_4] = [1, -1, 3]$

(c) $\begin{bmatrix} 2 & 2 & 4 & 3 & 2 \\ 0 & 0 & -1 & 11 & 1 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix} * [x_1, x_2, x_3, x_4, x_5] = [2, 0, 10]$

The examples above have no rows that are zero. What do we do when the matrix has some rows that are zero? Ignore them!

Consider an equation $\boldsymbol{a}_i \cdot \boldsymbol{x} = b_i$ where $\boldsymbol{a}_i = \boldsymbol{0}$.

- If $b_i = 0$ then the equation is true regardless of the choice of $\boldsymbol{x}$.
- If $b_i \neq 0$ then the equation is false regardless of the choice of $\boldsymbol{x}$.

Thus the only disadvantage of ignoring the rows that are zero is that the algorithm will not notice that the equations cannot be solved.

Problem 4: For each of the following matrix-vector equations, say whether the equation has a solution. If so, compute the solution.

(a) $$\begin{bmatrix} 1 & 3 & -2 & 1 & 0 \\ 0 & 0 & 2 & -3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * [x_1, x_2, x_3, x_4, x_5] = [5, 3, 2]$$

(b) $$\begin{bmatrix} 1 & 2 & -8 & -4 & 0 \\ 0 & 0 & 2 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * [x_1, x_2, x_3, x_4, x_5] = [5, 4, 0, 0]$$

Problem 5: Give a procedure `echelon_solve(row_list, label_list, b)` with the following spec:

- *input:* for some integer n, a matrix in echelon form represented by a list `row_list` of n vectors, a list of column-labels giving the order of the columns of the matrix (i.e. the domain of the vectors), and a length-n list $\boldsymbol{b}$ of field elements
- *output:* a vector $\mathbf{x}$ such that, for $i = 0, 1, \ldots, n-1$, the dot-product of `row_list`$[i]$ with $\mathbf{x}$ equals $\boldsymbol{b}[i]$ if `rowlist`$[i]$ is not a zero vector

Obviously your code should not use the `solver` module.

If you wanted to use this procedure with floating-point numbers, the procedure would have to interpret very small numbers as zero. To avoid this issue, you should assume that the field is $GF(2)$. (The interested student can modify the solution to work for $\mathbb{R}$.)

Hints for your implementation:

- The slickest way to write this procedure is to adapt the code of the procedure `triangular_solve(rowlist, label_list, b)` in module `triangular`. As in that procedure, initialize a vector `x` to zero, then iterate through the rows of `rowlist` from last row to first row; in each iteration, assign to an entry of `x`. In this procedure, however, you must assign to the variable corresponding to the column containing the first nonzero entry in that row. (If there are no nonzero entries in that row, the iteration should do nothing.)

  This approach leads to a very simple implementation consisting of about seven lines. The code closely resembles that for `triangular_solve`.

- For those for whom the above approach does not make sense, here is an alternative approach that leads to about twice as many lines of code. Form a new matrix-vector equation by removing

  - zero rows and
  - irrelevant columns,

  and then use the procedure `triangular_solve` in module `triangular` to solve the new matrix-vector equation.

  Removing zero rows: When you remove zero rows from the matrix, you must remove the corresponding entries from the right-hand side vector `b`. Because the matrix is in echelon form, all the zero rows come at the end. Find out which rows are zero, and form a new rowlist from the old that omits those rows; form a new right-hand side vector `b` by removing the corresponding entries.

  Removing irrelevant columns: For each of the remaining rows, find out the position of the leftmost nonzero entry. Then remove the columns that do not contain any leftmost nonzero entries.

Let $A\boldsymbol{x} = \boldsymbol{b}$ be the original matrix-vector equation, and let $\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ be the one resulting from these operations. Finally, solve $\hat{A}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}}$ using `triangular_solve`, and let $\hat{\boldsymbol{u}}$ be the solution. The domain of $\hat{\boldsymbol{u}}$ equals the column-label set of $\hat{A}$ rather than that of A. From $\hat{\boldsymbol{u}}$, construct a solution $\boldsymbol{u}$ to $A\boldsymbol{x} = \boldsymbol{b}$. The domain of $\boldsymbol{u}$ is the column-label set of A, and the extra entries of $\boldsymbol{u}$ (the ones not given by $\hat{\boldsymbol{u}}$) are set to zero (easy because of our sparsity convention).

Here are some examples to test your procedure.

'A'	'B'	'C'	'D'	'E'
one	0	one	one	0
0	one	0	0	one
0	0	one	0	one
0	0	0	0	one

  and b = [one, 0, one, one].

  The solution is

'A'	'B'	'C'	'D'	'E'
one	one	0	0	one

'A'	'B'	'C'	'D'	'E'
one	one	0	one	0
0	one	0	one	one
0	0	one	0	one
0	0	0	0	0

  and b = [one, 0, one, 0].

  The solution is

'A'	'B'	'C'	'D'	'E'
one	0	one	0	0

Problem 6: Now that you have developed a procedure for solving a matrix-vector equation where the matrix is in echelon form, you can use this in a procedure for the general case. We have already described the method. The code for the method is

```
def solve(A, b):
    M = echelon.transformation(A)
    U = M*A
    col_label_list = sorted(A.D[1])
    U_rows_dict = mat2rowdict(U)
    row_list = [U_rows_dict[i] for i in sorted(U_rows_dict)]
    return echelon_solve(row_list,col_label_list, M*b)
```

(Sorting is problematic when the column-labels include values of different types, e.g. ints and strings.)

Suppose you have the matrix

$A =$

	A	B	C	D
a	one	one	0	one
b	one	0	0	one
c	one	one	one	one
d	0	0	one	one

and the right-hand side vector $\boldsymbol{g}=$

a	b	c	d
one	0	one	0

You are interested in finding the solution to the matrix-vector equation $A\boldsymbol{x} = \boldsymbol{g}$. The first step in using Gaussian elimination to find the solution is to find a matrix M such that MA is in echelon form.

In this case, $M =$

	a	b	c	d
0	one	0	0	0
1	one	one	0	0
2	one	0	one	0
3	one	0	one	one

and $MA =$

	A	B	C	D
0	one	one	0	one
1	0	one	0	0
2	0	0	one	0
3	0	0	0	one

Use the above data and above procedure to figure out not the solution but what actual arguments should be provided to `echelon_solve` in order to obtain the solution to the original matrix-vector equation.

Finding a Basis for the Nullspace

Problem 7: We consider matrices over $GF(2)$. Let

$A =$

	A	B	C	D	E
a	0	0	0	one	0
b	0	0	0	one	one
c	one	0	0	one	0
d	one	0	0	0	one
e	one	0	0	0	0

.

Then the matrix

$M =$

	a	b	c	d	e
0	0	0	one	0	0
1	one	0	0	0	0
2	one	one	0	0	0
3	0	one	one	one	0
4	one	0	one	0	one

has the property that MA is a matrix in echelon form, namely

$MA =$

	A	B	C	D	E
0	one	0	0	one	0
1	0	0	0	one	0
2	0	0	0	0	one
3	0	0	0	0	0
4	0	0	0	0	0

List the rows $\boldsymbol{u}$ of M such that $\boldsymbol{u} * A = \mathbf{0}$. (Note that these are vectors in the null space of the transpose A^T.)

Problem 8: We consider matrices over $GF(2)$. Let

$A =$

	A	B	C	D	E
a	0	0	0	one	0
b	0	0	0	one	one
c	one	0	0	one	0
d	one	one	one	0	one
e	one	0	0	one	0

Then the matrix

$$M = \begin{array}{c|ccccc} & a & b & c & d & e \\ \hline 0 & 0 & 0 & \text{one} & 0 & 0 \\ 1 & 0 & 0 & \text{one} & \text{one} & 0 \\ 2 & \text{one} & 0 & 0 & 0 & 0 \\ 3 & \text{one} & \text{one} & 0 & 0 & 0 \\ 4 & 0 & 0 & \text{one} & 0 & \text{one} \end{array}$$

has the property that MA is a matrix in echelon form, namely

$$MA = \begin{array}{c|ccccc} A & B & C & D & E & \\ \hline 0 & \text{one} & 0 & 0 & \text{one} & 0 \\ 1 & 0 & \text{one} & \text{one} & \text{one} & \text{one} \\ 2 & 0 & 0 & 0 & \text{one} & 0 \\ 3 & 0 & 0 & 0 & 0 & \text{one} \\ 4 & 0 & 0 & 0 & 0 & 0 \end{array}$$

List the rows $\boldsymbol{u}$ of M such that $\boldsymbol{u} * A = \mathbf{0}$. (Note that these are vectors in the null space of the transpose A^T.)