6_systems

February 8, 2023

1 Systems

variationalform https://variationalform.github.io/

Just Enough: progress at pace https://variationalform.github.io/

https://github.com/variationalform

Simon Shaw https://www.brunel.ac.uk/people/simon-shaw.

This document uses python

and also makes use of LaTeX

in Markdown

1.1 What this is about:

You will be introduced to …

- simultaneous equations and their matrix representation as *systems*.
- the concept of *over* and *under* determined systems.
- the notion of matrix *rank*.
- using `numpy` to work with *systems of equations*.
- the concept of an *orthogonal* matrix.

As usual our emphasis will be on *doing* rather than *proving*: *just enough: progress at pace*

1.2 Assigned Reading

For this worksheet you should read Chapter 11 of [VMLS] for background to the linear algenra of vectors, and also Appendix D of [DSML] if you want to read more about `python` and `numpy`.

- VMLS: Introduction to Applied Linear Algebra - Vectors, Matrices, and Least Squares, by Stephen Boyd and Lieven Vandenberghe, https://web.stanford.edu/~boyd/vmls/
- DSML: Data Science and Machine Learning, Mathematical and Statistical Methods by Dirk P. Kroese, Zdravko I. Botev, Thomas Taimre, Radislav Vaisman, https://people.smp.uq.edu.au/DirkKroese/DSML/ and https://people.smp.uq.edu.au/DirkKroese/DSML/DSML.pdf

Further accessible material can be found in Chapter 1 of [FCLA] and in Chapters 2.3 of [MML].

- MML: Mathematics for Machine Learning, by Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. Cambridge University Press. `https://mml-book.github.io`.
- FCLA: A First Course in Linear Algebra, by Ken Kuttler, `https://math.libretexts.org/Bookshelves/Linear_Algebra/A_First_Course_in_Linear_Algebra_(Kuttler)`

All of the above can be accessed legally and without cost.

There are also these useful references for coding:

- PT: `python`: `https://docs.python.org/3/tutorial`
- NP: `numpy`: `https://numpy.org/doc/stable/user/quickstart.html`
- MPL: `matplotlib`: `https://matplotlib.org`

1.3 Linear Systems of Simultaneous Equations

The following type of problem occurs frequently across the whole landscape of applied mathematics - of which data science and analytics is a part. We'll see several examples of it as we go along. The problem is of this nature: find the values of x, y and z that *simultaneously* satisfy these three linear equations:

$$\begin{aligned} 3x - 2y + \ 4z &= \ 2 \\ -6x + 6y - 11z &= -1 \\ 6x + 2y + \ 5z &= \ 7 \end{aligned}$$

We can write this as: find $\boldsymbol{u} = (x, y, z)^T$ such that

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} \qquad \text{for } \boldsymbol{B} = \begin{pmatrix} 3 & -2 & 4 \\ -6 & 6 & -11 \\ 6 & 2 & 5 \end{pmatrix} \text{ and } \boldsymbol{f} = \begin{pmatrix} 2 \\ -1 \\ 7 \end{pmatrix}$$

If $\boldsymbol{B}$ is invertible then we can obtain $\boldsymbol{u}$, and hence x, y and z, by *solving the system*:

$$\begin{pmatrix} x \\ y \\ x \end{pmatrix} = \boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{f} = \frac{1}{36}\begin{pmatrix} 52 & 18 & -2 \\ -36 & -9 & 9 \\ -48 & -18 & 6 \end{pmatrix}\begin{pmatrix} 2 \\ -1 \\ 7 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -1 \end{pmatrix}.$$

What we see here is that if we have three unknowns, then we can determine them if we have three independent pieces of information. In this case that means the three equations. If those equations are not independent of each other then we wont be able to find all three unknowns.

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} \qquad \text{for } \boldsymbol{B} = \begin{pmatrix} 3 & -2 & 4 \\ -6 & 6 & -11 \\ 6 & 2 & 5 \end{pmatrix} \text{ and } \boldsymbol{f} = \begin{pmatrix} 2 \\ -1 \\ 7 \end{pmatrix}$$

Let's see this in `numpy`

[1]:
```
import numpy as np
B = np.array( [[3, -2, 4],[-6, 6, -11],[ 6, 2, 5 ]])
u = np.array([[2], [0], [-1]])
f = B.dot(u)
print('f = \n', f)
```

```
f =
 [[ 2]
 [-1]
 [ 7]]
```

[2]:
```
# here is how to solve a linear system like this...
u = np.linalg.solve(B, f)
print(u)
```

```
[[ 2.]
 [ 0.]
 [-1.]]
```

1.4 Under- and Over-Determined Systems

The point just made is quite subtle. Rather than explore the general theory let's look at some examples.

Example 1 The system, find $\boldsymbol{u}$ such that,

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} \qquad \text{for } \boldsymbol{B} = \begin{pmatrix} 3 & -2 & 4 \\ -6 & 6 & -11 \\ 6 & -4 & 8 \end{pmatrix} \text{ and } \boldsymbol{f} = \begin{pmatrix} 2 \\ -1 \\ 4 \end{pmatrix}$$

seems just like the one above, but there's an important difference. It's hard to spot, but if you look closely you can see that the third equation is exactly the same as the first multiplied by 2. Therefore these equations do not give three independent pieces of information - just two, and so there is not enough information to obtain three unknowns $\boldsymbol{u}$.

However, we can assert that $x = (10 - 2z)/6$ and $y = (9 + 9z)/6$ will solve this system for any choice of z (we'll see why in the lab).

Example 2 On the other hand, consider this problem: find $\boldsymbol{u}$ such that,

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} \qquad \text{for } \boldsymbol{B} = \begin{pmatrix} 3 & -2 & 4 \\ -6 & 6 & -11 \\ 9 & -8 & 15 \end{pmatrix} \text{ and } \boldsymbol{f} = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}.$$

Here you have to look even more closely: if you subtract the second equation from the first you'll get the third. So, again, there are three unknowns but only two (independent) equations.

Example 3 What do you make of this? Find $\boldsymbol{u} = (x, y)^T$ such that

$$\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f} \qquad \text{for } \boldsymbol{B} = \begin{pmatrix} 3 & -2 \\ -6 & 6 \\ 6 & 2 \end{pmatrix} \text{ and } \boldsymbol{f} = \begin{pmatrix} 6 \\ -18 \\ 7 \end{pmatrix}.$$

This is the same as

$$\begin{pmatrix} 3 & -2 \\ -6 & 6 \\ 6 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 6 \\ -18 \\ 7 \end{pmatrix}$$

or,

$$\begin{aligned} 3x - 2y &= 6 \\ -6x + 6y &= -18 \\ 6x + 2y &= 7 \end{aligned}$$

$$\begin{aligned} 3x - 2y &= 6 \\ -6x + 6y &= -18 \\ 6x + 2y &= 7 \end{aligned}$$

It turns out this is impossible: $x = 0$ and $y = -3$ is the unique (why?) solution to the first two equations yet using these in the third gives $2y = -6 \neq 7$.

There is therefore no solution to this problem because now we have three independent pieces of information (the equations), but only two degrees of freedom to fit that information.

Examples 1 and 2 are examples of *under-determined* systems. Here there are not as many equations as there are unknowns and when we remove the redundant equations the system matrices are short and wide (more columns than rows). We can expect to be able to solve such systems - although not uniquely.

Example 3 is an example of an *over-determined* system where there are more independent equations than unknowns. The system matrix is in this case tall and narrow (more rows than columns). These systems may not have solutions in the usual sense unless the extra equations are *consistent* with the rest of the system (although in that case it isn't really *over-determined*). The full story is complicated - here we're doing *just enough* so we can make *progress at pace*.

1.5 Rank

The rank (for our purposes) of a matrix corresponds to the number of independent equations it is capable of representing. In Examples 1 and 2 above the rank of the matrix, which is 2, is less than the dimension of the matrix, 3 in both cases.

When the rank of a square matrix is equal to its dimension, then that matrix has an inverse and we can solve the linear system in order to get a unique solution. This was the case in the original example above where we wrote $\boldsymbol{B}\boldsymbol{u} = \boldsymbol{f}$ and solved for $\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{f} = (2, 0, -1)^T$.

1.6 Orthogonal Matrices

Consider these two vectors,

$$\boldsymbol{u} = \frac{1}{5}\begin{pmatrix} 3 \\ 4 \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{v} = \frac{1}{10}\begin{pmatrix} 8 \\ -6 \end{pmatrix}.$$

It's easy to check that $\boldsymbol{u}^T\boldsymbol{v} = 0$ and so they are *orthogonal* (that is $\cos\theta = 0$ where θ is the angle between the vectors). Moreover, the denominators ensure that they are normalized to unit length, so they are in fact *orthonormal*. Let's stack them together horizontally and create a matrix like this,

$$\boldsymbol{Q} = (\boldsymbol{u}\ \boldsymbol{v}) = \begin{pmatrix} 3/5 & 8/10 \\ 4/5 & -6/10 \end{pmatrix} = \begin{pmatrix} 3/5 & 4/5 \\ 4/5 & -3/5 \end{pmatrix} = \frac{1}{5}\begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix}.$$

Now notice that because of the *orthonormality*,

$$\boldsymbol{Q}^T\boldsymbol{Q} = \frac{1}{25}\begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix}\begin{pmatrix} 3 & 4 \\ 4 & -3 \end{pmatrix} = \frac{1}{25}\begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \boldsymbol{I}$$

and so $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$. Such a matrix is called an *orthogonal* matrix. In this example $\boldsymbol{Q}$ is symmetric, but it need not be to have this property.

Indeed, if $\boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$, it will also follow that $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}$, even if $\boldsymbol{Q}$ is not symmetric.

This idea generalises to $n \times n$ matrices. We just have to imagine a matrix made by stacking a set of mutually orthogonal vectors next to each other. If each vector has n components, and there are n such vectors, then the matrix will be square. If the matrix is denoted by $\boldsymbol{Q}$ then it will follow that

$$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I} \qquad \text{the } n \times n \text{ identity matrix, and} \qquad \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}.$$

We will see orthogonal matrices below when we study the eigenvectors of symmetric matrices.

1.7 Review

We have made a lot of progress:

- we understand that linear equations can be written in matrix form.
- we saw how using `numpy` in `python` we could
  - represent linear systems using matrices and vectors;
  - create and solve linear systems with full rank;
- we understand the concept of an under-determined system
- we understand the concept of an over-determined system
- we know what is meant by an orthogonal matrix

We will be building extensively on these concepts. Matrices and vectors are form the branch of Mathematics called **Linear Algebra**.

Linear Algebra is of **central importance** in Data Science and Machine Learning.

1.8 Technical Notes, Production and Archiving

Ignore the material below. What follows is not relevant to the material being taught.

Production Workflow

- Finalise the notebook material above
- Clear and fresh run of entire notebook
- Create html slide show:
  - `jupyter nbconvert --to slides 6_systems.ipynb`
- Set `OUTPUTTING=1` below
- Comment out the display of web-sourced diagrams
- Clear and fresh run of entire notebook
- Comment back in the display of web-sourced diagrams
- Clear all cell output
- Set `OUTPUTTING=0` below
- Save
- git add, commit and push to FML
- copy PDF, HTML etc to web site
  - git add, commit and push
- rebuild binder

Some of this originated from

https://stackoverflow.com/questions/38540326/save-html-of-a-jupyter-notebook-from-within-the-r

These lines create a back up of the notebook. They can be ignored.

At some point this is better as a bash script outside of the notebook

[3]:
```
%%bash
NBROOTNAME='6_systems'
OUTPUTTING=1

if [ $OUTPUTTING -eq 1 ]; then
  jupyter nbconvert --to html $NBROOTNAME.ipynb
  cp $NBROOTNAME.html ../backups/$(date +"%m_%d_%Y-%H%M%S")_$NBROOTNAME.html
  mv -f $NBROOTNAME.html ./formats/html/

  jupyter nbconvert --to pdf $NBROOTNAME.ipynb
  cp $NBROOTNAME.pdf ../backups/$(date +"%m_%d_%Y-%H%M%S")_$NBROOTNAME.pdf
  mv -f $NBROOTNAME.pdf ./formats/pdf/

  jupyter nbconvert --to script $NBROOTNAME.ipynb
  cp $NBROOTNAME.py ../backups/$(date +"%m_%d_%Y-%H%M%S")_$NBROOTNAME.py
  mv -f $NBROOTNAME.py ./formats/py/
```

```
else
  echo 'Not Generating html, pdf and py output versions'
fi
```

```
Not Generating html, pdf and py output versions
```