Math Camp - Day 5

Aaron Rudkin

Trinity College Dublin
Department of Political Science

rudkina@tcd.ie

Inverse of a Matrix

- The inverse of a matrix A is the matrix A^{-1} that, when multiplied with A produces a $n \times n$ identity matrix
  $AA^{-1} = A^{-1}A = I_{n\times n}$
- Only square matrices can have inverses
- Only matrices with $det(A) \neq 0$ are invertible.

Inverse of a Matrix

- Inverse operation also called the **solution** of the matrix (In R, use `solve` command).
- It is possible, though annoying, to find the inverse of a matrix. Don't bother. Use a computer.
- Generic intuition: Create theoretical inverse $AA^{-1} = I_n$. We know A and I_n. Use algebra to solve equation:

$$\begin{bmatrix} 1 & -2 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$1 = 1a - 2c$$
$$0 = 1b - 2d$$
$$0 = 2a - 3c$$
$$1 = 2b - 3d$$

Use system of equations to solve:

$$\begin{bmatrix} -3 & 2 \end{bmatrix}$$

System of Equations Recap

- Substitution: Solve for one variable in terms of others, substitute in equation, keep solving

  Upside: Easy; Downside: Not very systematic!

System of Equations Recap

- Substitution: Solve for one variable in terms of others, substitute in equation, keep solving

  Upside: Easy; Downside: Not very systematic!

Given:

$$
\begin{aligned}
1 &= 1a - 2c \\
0 &= 1b - 2d \\
0 &= 2a - 3c \\
1 &= 2b - 3d
\end{aligned}
$$

System of Equations Recap

- Substitution: Solve for one variable in terms of others, substitute in equation, keep solving

  Upside: Easy; Downside: Not very systematic!

Solve:

Given:

$$1 = 1a - 2c$$
$$0 = 1b - 2d$$
$$0 = 2a - 3c$$
$$1 = 2b - 3d$$

$a = 1 + 2c$	Solve for a
$2(1+2c) - 3c = 0$	Substitute
$c = -2$	Solve for c
$a = 1 + 2(-2) = -3$	Solve for a
$b = 2d$	Solve for b
$2(2d) - 3d = 1$	Substitute
$d = 1$	Solve for d
$b = 2(1) = 2$	Solve for b

System of Equations Recap

- Elimination: Subtract a multiple of one equation from another to eliminate a term

Given:

$$\begin{aligned}1 &= 1a - 2c\\0 &= 1b - 2d\\0 &= 2a - 3c\\1 &= 2b - 3d\end{aligned}$$

System of Equations Recap

- Elimination: Subtract a multiple of one equation from another to eliminate a term

Solve:

Given:

$$1 = 1a - 2c$$
$$0 = 1b - 2d$$
$$0 = 2a - 3c$$
$$1 = 2b - 3d$$

$$\begin{aligned} 2(1a - 2c &= 1) \\ -(2a - 3c &= 0) \rightarrow c = -2 \\ (0a - 1c &= 2) \end{aligned}$$

$$\begin{aligned} 2(1b - 2d &= 0) \\ -(2b - 3d &= 1) \quad \rightarrow d = 1 \\ (0b - 1d &= -1) \end{aligned}$$

Other Solutions for Inverses

- **Cramer's Rule**: $A^{-1} = \frac{1}{det(A)} adj(A)$
- $adj(a) = C'$ where C is the cofactor matrix and C' is its transpose
- A cofactor matrix is the matrix of all cofactors, and recall a cofactor is the term:

  $(-1)^{(i+j)} M_{ij}$
- And recall that M_{ij} is the minor (the determinant of the matrix after removing one row and one column)
- Disadvantage: Slow even for computers! Requires taking many, many determinants and many, many submatrices

Other Solutions for Inverses

- **Eigendecomposition** This is the method computers use to solve matrix inverses.
- Too much linear algebra to elaborate, but computers can quickly "factor" a matrix into "eigenvalues" and "eigenvectors"
- Plain English: "eigenvectors" (v) are the underlying structure of the data rotated... and "eigenvalues" (λ) are scalar stretch factors or weights.
- The inverse of the original matrix, $A^{-1} = Q \times \Lambda^{-1} \times Q^{-1}$
- Λ is diagonal matrix of eigenvalues: inverse of a diagonal matrix is $b_{ii} = \frac{1}{a_{ii}}$
- Q is eigenvector matrix – for reasons left unexplained, Q is a special matrix where $Q^{-1} = Q'$
- Turns problem of inverse (hard) into problem of eigendecomposition (easy), transpose (easy), and scalar division (easy) – way easier for computers!

Other Solutions for Inverses

- There are dozens of other solutions for matrix inverses: **blockwise inversion**, Newton's Method, LU decomposition, Cayley-Hamilton
- Blockwise inversion: Matrix inverses for large matrices are hard, but you can break a matrix into chunks (say, quarters) and take the inverse for each chunk and use that to calculate larger inverse.
- Every method of matrix inversion is quite computationally intensive
- Matrix inversion of medium to large matrices impossible before computers
- Means we couldn't actually solve linear regression for large amounts of data or predictors before computers
- No reason for you to take inverse on your own, just understand intuition of what an inverse is and how it fits into matrix algebra

Matrix Inverse Properties

Inverse	$(A^{-1})^{-1} = A$
Multiplicative property	$(AB)^{-1} = B^{-1}A^{-1}$
Scalar multiplication $(n \times n)$	$(cA)^{-1} = c^{-1}A^{-1}$ if $c \neq 0$
Transpose	$(A^{-1})^T = (A^T)^{-1}$

Linear Independence and Matrix Rank

Linear Independence

- Given a set of vectors $v : v_1, v_2, v_3, ...v_n$, the vectors can be said to be **linearly dependent** (**collinear**) or **linearly independent**
- The vectors are linearly dependent if some combination of the vectors results in the zero vector and independent if no such combination exists
- Formally, the math is that vectors are linearly independent if no solution to $a_1 v_1 + a_2 v_2 + ... + a_n v_n = 0$ except for $a_1 = a_2 = ... = a_n = 0$

Linear Independence Examples

- $a = (1,3,5),\ b = (3,9,15) \rightarrow$ NOT INDEPENDENT.
- Setting the *a* weight to -3 and the *b* weight to 1 results in a 0 vector.
- $$-3 \times a + 1 \times b = \begin{pmatrix} -3+3 \\ -9+9 \\ -15+15 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- $c = (2,2,0),\ d = (1,-1,1) \rightarrow$ Linearly independent
- Any weight on d besides 0 would leave the last dimension of d intact (e.g. $0 + \hat{v}_2 \times 1 \neq 0$)
- But setting the weight 0 on d would leave the first and second dimensions of intact unless c's weight is 0: (e.g. $\hat{v}_1 \times 2 + 0 \times 1 \neq 0$)
- So then the only solution is 0s for all weights, which means linear independence

Matrix Rank

- The **rank** of a matrix is defined as (a) the maximum number of linearly independent column vectors in the matrix or (b) the maximum number of linearly independent row vectors in the matrix. Both definitions are equivalent.
- The maximum rank of a matrix is the smaller of its two dimensions, for an $i \times j$ matrix;
  - If i is less than j, then the maximum rank of the matrix is i.
  - If i is greater than j, then the maximum rank of the matrix is j.
- Rank of A and B (hint for B: column 1 + column 2)

$$A = \begin{pmatrix} 1 & 2 & 1 & 3 \\ 2 & 3 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 0 & 2 \\ 3 & 1 & 4 \\ 1 & 4 & 5 \\ 3 & 3 & 6 \end{pmatrix}$$

Matrix Rank through Gauss-Jordan Elimination

- Not every example is as clear as those above; need systematic way to find the rank of a matrix.
- Normal way is through **Gauss-Jordan Elimination**
  - Goal: Make matrix upper triangular by doing basic operations
  - Multiply rows
  - Add rows to another
  - Swap rows
- Resulting Matrix is **row-echelon form**: number of rows that aren't all 0s is rank of matrix

Example

Goal: Make Matrix upper triangular.

$$B = \begin{pmatrix} -3 & 2 & -1 & -1 \\ 6 & -6 & 7 & -7 \\ 3 & -4 & 4 & -6 \end{pmatrix}$$

Current step: Eliminate row 2, column 1.

Method: Add $2\times$ row 1 to row 2.

$$\begin{pmatrix} -6 & 4 & -2 & -2 \end{pmatrix}$$

Example

Goal: Make Matrix upper triangular.

$$B = \begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 3 & -4 & 4 & -6 \end{pmatrix}$$

Current step: Eliminate row 3, column 1.

Method: Add row 1 to row 3.
$\begin{pmatrix} -3 & 2 & -1 & -1 \end{pmatrix}$

Example

Goal: Make Matrix upper triangular.

$$B = \begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{pmatrix}$$

Current step: Eliminate row 3, column 2.

Method: Add $-1\times$ row 2 to row 3.
$\begin{pmatrix} 0 & 2 & -5 & 9 \end{pmatrix}$

Example

$$B = \begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{pmatrix}$$

Goal: Make Matrix upper triangular.
Current step: We've done it!

How many rows are non-zero? 3. B is full rank, rank $= 3$

Reasons for Lower Rank: Multicollinearity

- We call a matrix **full rank** if its rank is its maximum rank: in other words, if every row/column in the matrix is independent.
- In the simplest case, like A on the previous slide, linear dependence means that one row (or column) is a multiple of another.This might happen if you have two explanatory variables which measure the same data in a different scale (e.g. GDP in Euro, GDP in USD)
- In a more complex case, like B on the previous slide, this can mean that one row (or column) is a combination of multiples of other rows or columns. This might happen if you have an explanatory value that is made up of components that are themselves explanatory variables.
- This is called **perfect multicollinearity**
- Non-math takeaway: When two explanations are perfectly collinear, it is impossible for the model to know which to "blame" for outcome.

Reasons for Lower Rank: $n < p$

- OLS cannot be estimated when the number of observations n is less than the number of explanatory variables p.
- Why is this? Think about your data as a matrix X, where rows are observations and columns are explanatory variables ($n \times p$ matrix)
- The math behind OLS, as we'll see later on, involves the term $(X'X)^{-1}$
- If X is $(n \times p)$, then X' is $(p \times n)$ and X'X will be $(p \times p)$
- Assuming no perfect multicollinearity, the rank of X will be at most n when $n < p$.
- So the rank of the resulting $X'X$ is at most n, although its full rank will be p
- So $X'X$ has no inverse, and we can't solve OLS
- What to do in this case? Two options: (1) change the data – simplify model or gather more observations; (2) fit a penalized model, which can get around the rank restriction

Multivariate Calculus

Functions of Several Variables

- We've seen functions of more than one variable before:

$$f(x, y, z)$$

- In the past we've talked about scalar outputs, where the answer is a number

$$f(x, y, z) = x^2 + xy^2 - z^3$$

- Vector outputs are also possible, with one equation for each output:

$$f(x, y, z) = \begin{pmatrix} x^2y - z \\ y^3z^2 - x^2 \\ \frac{3x^3z^2}{y^2} \end{pmatrix}$$

Partial Derivatives

- We ran into partial derivatives before now; when we take the derivative of a multivariable equation with respect to one of the variables.
- Say we have the function $f(x, z) = xz$. The partial derivatives reflect the instantaneous rates of change in y due to the variable we differentiate, holding the other constant:

$$\frac{\delta y}{\delta x} xz = zdx \qquad \frac{\delta y}{\delta z} xz = xdz$$

- In other words, the partial derivative is linked intrinsically to "marginal effects", substantively. A marginal effect asks how a certain factor (education, salary, vaccination, training, audits) can effect an outcome of interest; but that outcome is also driven by other factors, and so we wish to "hold the other variables constant" while manipulating or observing variation in our variable of interest.

Gradient Vector

- **Gradient vector** is a special vector that includes all of the partial derivatives of a system of equations.
- $f(x, y, z) = 4x + 6y - z$
  - $\nabla f = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ -1 \end{pmatrix}$
- Solving optimization problems of multiple variables that do not have analytic solutions – so we can't just find a global minimum/maximum – often requires using "gradient descent": using a computer to try to climb down the gradient, numerically.

Jacobian Matrix

- Jacobian matrix —**J**—is something of an extension; given i equations of k variables, the Jacobian is the matrix that contains every partial derivative of every equation
- Each row has the partial derivatives with respect to each variable of a single equation
- Each column has the partial derivatives with respect to one variable of every equation
- N_{row} = The number of functions and N_{column} =The number of unknowns

$$\begin{aligned} f_1 &= x^2 + 3xy - z^3 \\ f_2 &= xy^2 - xz^2 \\ f_3 &= xyz - xy^4 + z^2 \\ f_4 &= x + y + z \end{aligned} \qquad J = \begin{pmatrix} 2x+3y & 3x & 3z^2 \\ y^2 - z^2 & 2xy & -2xz \\ yz - y^4 & xz - 4xy^3 & 2z \\ 1 & 1 & 1 \end{pmatrix}$$

Hessian Matrix

- The **Hessian** matrix is a square matrix of every second-order partial derivatives of a scalar-valued function
- By second-order, we mean taking the derivative with respect to each set of two variables (e.g. for a two-variable system: (x, x), (x, y), (y, x), and (y, y))

$$\begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Note syntax for the second-order "cross partials": taking a partial derivative on one variable at a time in sequence.

The Shape of Things to Come: How OLS works

- If you have not already, you will still encounter the magic of **OLS**, ordinary least squares, the hammer in your toolkit for fitting linear models.
- Given a vector of outcome variables y and a matrix of predictors X, OLS posits the following data generating mechanism:

  $$y = X\beta + \epsilon$$
- In other words, data y $(n \times 1)$ can be explained by assigning weights β $(p \times 1)$ to each of the predictors in X $(n \times p)$, and the data also contains a stochastic component ϵ $(n \times 1)$ that cannot be explained.
- How do we choose the weights $\hat{\beta}$? We want to choose the best weights $\hat{\beta}$ (the hat implies our "best estimate") to minimize the sum of squared errors.

Solving OLS $\hat{\beta}$

- First, let's solve the data generating process for the error term:

  $$e = y - X\hat{\beta}$$

- Note we've swapped from ϵ (the underlying and unobservable, unexplainable stochasticity in the data) to e, the portion of the variation unexplained by our model but observable.
- Rather than optimize the error, we optimize the sum of squared errors; square both sides (left-multiply by transpose):

  $$e'e = (y - X\hat{\beta})'(y - X\hat{\beta})$$

- Notice that $e'e$ is a $(1 \times n) \times (n \times 1)$ matrix – in other words, a 1 by 1 scalar.
- Let's proceed using our algebra rules to expand the binomials above...
- Remember $(AB)' = B'A'$ and $(A + B)' = A' + B'$

  $$e'e = y'y - y'X\hat{\beta} - \hat{\beta}'X'y + \hat{\beta}'X'X\hat{\beta}$$

- Here we've used FOIL.

Solving OLS $\hat{\beta}$

- From this position, outside term gives us $y'X\hat{\beta}$ and inside term gives us $\hat{\beta}'X'y$:
$$e'e = y'y - y'X\hat{\beta} - \hat{\beta}'X'y + \hat{\beta}'X'X\hat{\beta}$$
- We can combine these into one term
$$e'e = y'y - 2\hat{\beta}'X'y + \hat{\beta}'X'X\hat{\beta}$$
- Why does this combination work?
- y is $(n \times 1)$, X is $(n \times p)$, and $\hat{\beta}$ is $(p \times 1)$, so:
  - $y'X\hat{\beta}$ is $(1 \times n)\ (n \times p)\ (p \times 1) = 1\text{x}1 =$ a scalar
  - $\hat{\beta}'X'y$ is $(1 \times p)\ (p \times n)\ (n \times 1) = 1\text{x}1 =$ a scalar
  - The transpose of a scalar is the same scalar, so $y'X\hat{\beta} = (y'X\hat{\beta})' = \hat{\beta}'X'y$

Solving OLS $\hat{\beta}$

- From our FOIL expansion:
$$e'e = y'y - 2\hat{\beta}'X'y + \hat{\beta}'X'X\hat{\beta}$$
- Now we should take the derivative with respect to $\hat{\beta}$ and set the FOC to find the local minima
$$\frac{\delta}{\delta\hat{\beta}}e'e = -2X'y + 2X'X\hat{\beta} = 0$$
- Yes, matrices and vectors can use calculus, and so we can take their derivatives! A little tricky, but still... let's continue:
$$2X'X\hat{\beta} = 2X'y$$
$$X'X\hat{\beta} = X'y$$

Solving OLS $\hat{\beta}$

- From our FOC:
  $$X'X\hat{\beta} = X'y$$
- We want to isolate $\hat{\beta}$ on left hand side, which means we need to choose some operation that can get rid of $X'X$.
- From rules: Given a matrix A, $AA^{-1} = I = A^{-1}A$.
- Recall I is the identity matrix, and multiplying anything by it does not affect the result.
- Thus we can left-multiply both sides by $(X'X)^{-1}$:
  $$(X'X)^{-1}(X'X)\hat{\beta} = (X'X)^{-1}X'y$$
- Inverse and term cancel each other out, yielding I, which gives us a solution:
  $$I\hat{\beta} = \hat{\beta} = (X'X)^{-1}X'y$$
- And now we've solved OLS's best estimates.

Extra Properties of OLS: Uncorrelated errors

- Some interesting properties come from this process.
- Take the **normal form** equation:
$$X'X\hat{\beta} = X'y$$
- Substitute y for $X\hat{\beta} + e$ – after all, that's the definition of our model:
$$X'X\hat{\beta} = X'(X\hat{\beta} + e)$$
- But wait a minute, this implies...
$$X'e = X'X\hat{\beta} - X'X\hat{\beta} = 0$$
- Think of the term $X'e$ as the correlation between our predictors X and our errors e: it's 0
- Short takeaway: $X'e = 0$ because OLS explained as much relationship between X and Y as possible. Everything left in the error is uncorrelated with X.
- Other, harder takeaway: Average error will be 0 (OLS never aims too high or too low)!

Next Steps

The last few slides are advanced material! This is the kind of proof that takes a while to learn. The important thing is to understand that the building blocks we've covered: algebra, derivatives, and matrices come together to solve an important problem. If we want to learn the relationships to be found in our data – the essence of data science and quantitative reasoning – these are the skills we'll need.

Thank you for your patience, and enjoy your semester!