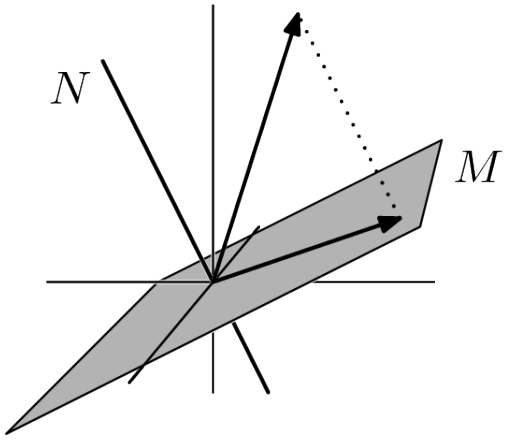

Hanbat National University

Course: Linear Algebra

January 5, 2023

Author	*Student ID*
Jonghyeok Kim	20201967

Linear Algebra (1/8)

Jonghyeok Kim

January 5, 2023

Contents

1 Linear Equations and Matrices **4**

1.1 Introduction to Linear Algebra 4

1.1.1 System of linear equations 4

1.1.2 Types of solutions 4

1.2 Gaussian Elimination 5

1.2.1 Introduction to matrices 5

1.2.2 Elementary row operations 5

1.2.3 Gaussian elimination 6

1.2.4 Extending row operations 7

1.3 The Transpose and Inverse of a Matrix 7

1.3.1 Transpose of a matrix 8

1.3.2 Properties of matrix tranpose 9

2 Conclusion **10**

2.1 Conclusion 10

3 Acknowledgement **10**

A Types of Matrices **11**

A.1 Tutorial Overview 11

A.1.1 Square Matrix 11

A.1.2 Symmetric Matrix 12

A.1.3 Triangular Matrix 13

B References **16**

1 Linear Equations and Matrices

1.1 Introduction to Linear Algebra

Linear Algebra는 Linear Equations들에 대한 Study이다.

여기서 Linear Equation이란, 모든 변수들 (ex. $x, y, z, \cdots$)이 0 또는 1의 지수를 가지는 것이다.

만약에 어떤 방정식이 Trigonometric, Exponential, Logarithmic 또는 Hyperbolic 함수들의 Argument를 포함하고 있다면 이 경우 Linear하다고 하지 않는다. (ex. $sin(x) + y + z = 1$)

1.1.1 System of linear equations

참고로 Linear Equation의 집합을 System of Linear Equations 또는 Linear System이라 한다. 단, Linear Equations들의 수가 유한해야 하고, 유한 개의 미지수를 가져야 한다.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \quad\quad \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{1}$$

식 (1)에서 a_{ij}, b_i는 실수(real number)이고, $x_1, x_2, x_3, \cdots, x_n$은 실수에 대한 Placeholder이다.

1.1.2 Types of solutions

선형대수학(linear algebra)는 Section 1.1에서 상술하였듯, Linear Equations들에 대한 Study이므로, 이러한 Linear Equations들에 대한 해(solution)가 존재하는지/존재하지 않는지, 만약 존재한다면 얼마나 많은 해가 존재하는지에 대한 근본적인 의문들을 가진다.

만약 Linear System이 해를 가지지 않는다면, **inconsistent**하다고 말한다.

반면에, Linear System이 최소한 하나 이상의 해를 가진다면, **consistent**하다고 말한다.

해(solution)는 다음과 같이 크게 세 가지 종류로 구분할 수 있다.

- No solution
- Unique solution
- Infinite number of solution

1.2 Gaussian Elimination

1.2.1 Introduction to matrices

다음과 같은 Linear system이 있다고 해보자.

$$\begin{cases} x + 2y = 3 \\ 2x + y = 2.5 \end{cases} \tag{2}$$

위의 식 (2)의 좌변의 첫 번째 열(column)은 미지수 x의 계수들만이 존재하고, 두 번째 열은 미지수 y의 계수들만이 존재한다.

그러므로, 이러한 미지수의 계수들을 다음과 같은 Matrix로 표현할 수 있다.

$$\begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix} \tag{3}$$

또한, 식 (2)의 등호 기준 우변에는 좌변과 달리 하나의 열(column)만이 존재하기 때문에 이를 다음과 같이 하나의 벡터(vector)로 나타낼 수 있다.

$$\begin{pmatrix} 3 \\ 2.5 \end{pmatrix} \tag{4}$$

참고로 Vector는 Matrix의 한 종류이다. 즉, Vector는 Matrix이다.

이렇듯 Matrix는 데이터를 저장하기 위한 효율적인 방법이다.

1.2.2 Elementary row operations

앞서 살펴본 식 (1)은 Section 1.2.1에서 다룬 Matrix를 이용하여 다음과 같이 표현할 수 있다.

$$\left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right) \tag{5}$$

위의 Matrix (5)는 Augmented matrix인데, 수직선 기준 좌변은 미지수 $x_1, x_2, x_3, \cdots, x_n$의 계수(coefficients)들이 오고, 우변은 Linear equation의 등호 기준 우변에 있는 상수(constant) 값이 온다. 즉, 앞서 Section

1.2.1에서 살펴본 Matrix (3)와 (4)를 하나의 Matrix로 표현한 것이 바로 이 Augmented Matrix이다.

여기서, Augmented란 'To increase'를 의미하는데, 다시 말해서 '증가되어진', '추가되어진'이라는 의미로 직역할 수 있다.

즉, 기존 Matrix가 다음과 같이 표현되는데 반해,

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix} \tag{6}$$

Augmented matrix는 위의 기존 Matrix (6)에 '수직선(vertical line)'이 추가되어진 것이다.

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{array}\right) \tag{7}$$

이러한 Augmented matrix는 이전에 살펴보았던 Matrix (5)에서와 같이 어떤 Linear system of equations을 짧고 간략하게 표현하는 데 사용된다.

1.2.3 Gaussian elimination

다음과 같은 Linear system이 있다고 해보자.

$$\begin{cases} x - 3y + 5z = -9 \\ 2x - y - 3z = 19 \\ 3x + y + 4z = -13 \end{cases} \tag{8}$$

식 (8)을 다음과 같이 변환하는 과정을 Gaussian Elimination with Back substitution이라 한다.

$$\begin{cases} x - 3y + 5z = -9 \\ 5y - 13z = 37 \\ 15z = -60 \end{cases} \tag{9}$$

어떻게 보면, Lower triangular matrix 즉 Main diagonal 기준, Bottom part를 0으로 만드는 과정으로 생각할 수도 있겠다.

1.2.4 Extending row operations

최종적으로, 다음과 같은 Augmented Matrix로 변환하는 것이 목표이다.

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & * \\ 0 & 1 & 0 & * \\ 0 & 0 & 1 & * \end{array}\right) \tag{10}$$

위와 같은 Augmented matrix를 Reduced row echelon 형태에 있다고 한다.

어떠한 행렬이 Reduced row echolon 형태에 있기 위해서는 다음의 조건들을 모두 만족시켜야 한다.

1. 0만을 포함하고 있는 행들이 Matrix의 Bottom part에 위치해 있어야 한다.
2. 어떤 행이 0이 아닌 Entries들을 가지고 있는 데, 첫 번째로 오는 0이 아닌 Entry는 1이어야 한다. (이때 1을 Leading 1이라 한다.)
3. 두 개의 연이은 0이 아니고, Leading 1의 Entries를 가지고 있는 행들이 Top left $\sim$ Bottom right상에 위치해 있어야 한다.
4. Leading 1을 포함하고 있는 어떤 열에서 0이 아닌 Entry가 오직 Leading 1 뿐이어야 한다.

위의 조건들 중 마지막 4번의 조건이 만족되지 않고 나머지 조건들만 만족된다면, 이 경우 해당 Augmented matrix가 Row echelon 형태에 있다고 한다.

참고로, 조건에서 상술하였듯 첫 번째로 오는 1의 Entry를 두고 Leading 1이라 하는데, 일부 선형대수학 문헌에서는 0이 아닌 어떠한 Leading 숫자들에 대해 Leading coefficient라는 용어를 사용하고 있다.

만약 어떠한 Augmented Matrix를 Row echolon 형태로 변환하는 과정을 Gaussian Elimination이라 하고,

Reduced row echolon 형태로 변환하는 과정은 Gauss-Jordan elimination이라 한다.

결론적으로, Augmented Matrix를 Reduced row echolon 형태로 변환하는 목적은 Back substitution을 피하기 위해서이다.

추후에 증명하겠지만, 만약 어떠한 Augmented matrix가 주어진다면, 그것의 Reduced row echelon 형태는 유일하나, Row echelon 형태는 유일하지 않다.

1.3 The Transpose and Inverse of a Matrix

Matrix는 실수(Real number)에서와 같이 나누기 연산이 불가능하지만, 이것과 가장 유사한 연산으로서 **Inverse** 연산이 존재한다.

이러한 Inverse Matrix를 통해 Linear Systems의 해를 찾는 것이 가능하다.

뒤에서도 살펴보겠지만, Inverse 연산 외에 또 다른 중요한 연산으로서 **Transpose** 연산이 존재한다.

1.3.1 Transpose of a matrix

Transpose 연산은 주어진 Matrix의 행(Rows)을 회전시킨 새로운 Matrix를 만들어주는 연산이다.

예를 들어, 다음과 같은 Matrix가 있다고 해보자.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \tag{11}$$

위 Matrix (11)의 Transpose는 다음과 같다.

$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} \tag{12}$$

위 예제를 통해 확인할 수 있듯이, Column 1은 Row 1이 되고, Column 2는 Row 2가 된다.

특히, A의 Transpose는 A^T로 표기할 수 있다.

일반적으로, Matrix A의 entry인 a_{ij}는 A^T 내의 entry인 a_{ji}로 Transpose 되어진다.

앞서 Linear systems의 미지수(Unknown)의 벡터(Vector)인 $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$는 Column 벡터로서, 이것을 Transpose 하게 되면, Row 벡터가 되므로, 이를 Row 벡터의 Transpose로 다음과 같이 나타낼 수 있다.

$$\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}^T \tag{13}$$

이렇듯 Column 벡터를 Row 벡터로 표기하는 것은 공간을 절약할 수 있는데, 위 경우에는 Column 벡터 $\mathbf{x}$가 n 개의 Lines을 차지하는 데 반해, $\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}^T$ 는 1개의 Line만을 사용할 수 있다.

중요한 것은 $m \times n$의 Matrix를 Transpose하게 되면, $n \times m$의 Matrix가 도출된다는 사실이다.

만약 $n \neq m$이면, Tranpose 연산으로 Matrix의 모양(shape)을 변화시킬 수 있다.

1.3.2 Properties of matrix tranpose

Theorem 1. *Properties of matrix transpose*

A와 B가 아래의 연산(Operations)들이 수행될 수 있도록 적절한 크기(An appropriate size)를 가진 Matrices라고 해보자. 여기서, k는 스칼라(Scalar)이다.

(a) $(A^T)^T = A$

(b) $(kA)^T = kA^T$

(c) $(A + B)^T = A^T + B^T$

(d) $(AB)^T = B^T A^T$

위 Theorem 1 중 (a)의 의미는 다음과 같다.

Figure 1: The meaning of $(A^T)^T$

즉, Transpose 처리된 Matrix를 다시 Transpose 연산을 적용하게 되면, 처음 시작했던 Matrix가 된다는 의미이다.

이를 Top left에서 Bottom right까지의 Diagonal에 대해 Matrix를 뒤집은 다음, 다시 Diagonal에 대해 뒤집으면, 처음의 Matrix가 다시 도출되는 것으로 생각해볼 수 있다.

이를 증명해보자.

Proof 1

a_{ij}를 Matrix A의 i번째 행과 j번째 열에 있는 Entry라 하자.

그러면 이는 다음과 같다.

$$A^T = (a_{ij})^T = (a_{ji}) \tag{14}$$

위의 식 (14)에 대해 또 다시 Transpose를 취하게 되면 다음과 같다.

$$(A^T)^T = (a_{ji})^T = (a_{ij}) = A \ \square \tag{15}$$

위 Proof 1에서 Entries들은 두 번 뒤바뀌었는데, 예를 들어, $a_{21} \to a_{12} \to a_{21},\ a_{31} \to a_{13} \to a_{31}$ 이다.

그러므로, $(A^T)^T = A$임이 증명되었다.

이번에는 Theorem 1 중 (c)를 증명해보자.

Proof 2

$$
\begin{aligned}
(A+B)^T &= \left[\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix} \right]^T \\
&= \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{pmatrix}^T \\
&= \begin{pmatrix} a_{11}+b_{11} & a_{21}+b_{21} & \cdots & a_{m1}+b_{m1} \\ a_{12}+b_{12} & a_{22}+b_{22} & \cdots & a_{m2}+b_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n}+b_{1n} & a_{2n}+b_{2n} & \cdots & a_{mn}+b_{mn} \end{pmatrix} \\
&= \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{21} & \cdots & b_{m1} \\ b_{12} & b_{22} & \cdots & b_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ b_{1n} & b_{2n} & \cdots & b_{mn} \end{pmatrix} = A^T + B^T \ \square
\end{aligned}
\tag{16}
$$

2 Conclusion

2.1 Conclusion

3 Acknowledgement

Appendix A

A Types of Matrices

A.1 Tutorial Overview

Matrices는 크게 다음과 같이 6가지 종류로 나뉘어진다.

1. Square Matrix
2. Symmetric Matrix
3. Triangular Matrix
4. Diagonal Matrix
5. Identity Matrix
6. Orthogonal Matrix

위의 Matrix 중 4번을 제외한 모든 Matrix는 1번의 Square Matrix에 포함된다. (즉, Square Matrix이다.)

A.1.1 Square Matrix

Square Matrix는 행(Rows) n의 수와 열(Columns) m의 수가 동일한 Matrix이다.

$$n \equiv m \tag{17}$$

이러한 Square Matrix는 행(Rows)과 열(Columns)의 수가 동일하지 않은 Rectangular Matrix와는 대조된다.

Square Matrix의 차원은 보통 n을 이용하여 표기되는 데, 예를 들면 다음과 같은 형식으로 표기된다.

$$n \times n \tag{18}$$

또한, Matrix의 크기를 Order라고 하는 데, 예를 들어, Order 4의 Square Matrix는 4×4로 표기된다.

즉, Square Matrix는 행(Rows)과 열(Columns)이 동일하기 때문에 Matrix의 크기 Order 4가 하나만 주어지면 나머지 하나의 정보를 알 수 있다.

특히, Square Matrix의 Top left에서 Bottom right까지의 Diagonal상의 값들에 대한 Vector를 Main diagonal 이라 부른다.

예를 들어, 아래와 같은 Square Matrix가 있다고 하면,

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \tag{19}$$

이때 Main diagonal(Main diagonal은 정의에서 이미 확인한 것처럼, Vector이다.)은 다음과 같다.

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \tag{20}$$

이러한 Square Matrix는 서로 손쉽게 더해지고 곱해질 수 있으며, 회전(Rotations)(e.g. 이미지 회전)과 같은 많은 단순한 선형 변환(Linear transformation)의 기초(Basis)가 된다.

A.1.2 Symmetric Matrix

Symmetric Matrix는 Square Matrix의 한 종류로서, Top-right Triangle과 Bottom-left Triangle이 동일한 Matrix이다.

Introduction to Linear Algebra Fifth Edition(2016)의 Symmetric Matrices에 대한 다음의 설명을 참조해보자.

> “It is no exaggeration to say that symmetric matrices S are the most important matrices the world will ever see - in the theory of linear algebra and also in the applications.”

직역해보면, 선형대수학의 이론과 실제 응용에 있어서 전세계를 통틀어 Symmetric Matrices가 가장 중요한 Matrices라고 말하는 것은 과장되지 않았다는 의미이다. 즉, 그만큼 Symmetric Matrices가 중요한 Matrices라는 의미이다.

어떤 Matrices가 Symmetric이 되기 위해서는 대칭축(The axis of symmetry)이 항상 그 Matrix의 Main diagonal(Top left $\sim$ Bottom right)이어야 한다.

즉, Main diagonal이라는 용어는 Square Matrices에서 정의되므로, 결국 Symmetric Matrices는 이러한 Square Matrices의 한 종류임을 유추해볼 수 있다.

이러한 Symmetric Matrices의 예를 보이면 다음과 같다.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 \\ 4 & 3 & 2 & 1 & 2 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix} \tag{21}$$

위의 Matrix (21)를 보면, Main diagonal 기준(대칭축)으로 그 Entries들이 서로 동일함을 확인해볼 수 있다. 즉, 해당 Matrix는 Symmetric Matrix이다.

어떠한 Symmetric Matrix가 있다면, 그것은 항상 Square하고 그 자신의 Transpose와 동일하다.

여기서 Transpose란 행과 열의 수를 뒤바꾸는 즉 그 위치를 서로 뒤바꾸는 연산(Operations)이다.

$$M = M^T \tag{22}$$

A.1.3 Triangular Matrix

Triangular Matrix란 Square Matrix의 한 종류인데, 그 Matrix의 Upper-right 또는 Lower-left에서의 모든 값들이 0의 값으로 채워진 Matrix이다.

Main diagonal의 위쪽에만 값(0이 아닌 값)이 있는 Triangular Matrix를 Upper-triangular Matrix라 하고, 반대로 Main diagonal의 아랫쪽에만 값(0이 아닌 값)이 있는 Triangular Matrix를 Lower-triangular Matrix라 한다.

즉, 이러한 Triangular Matrix 역시 그 정의상에서 Square Matrix의 정의에 등장하는 Main diagonal이라는 개념이 수반되었으므로, 이 역시 Square Matrix임을 유추해볼 수 있다.

아래는 3×3의 Upper-triangular Matrix의 한 예제이다.

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{pmatrix} \tag{23}$$

아래는 3×3의 Lower-triangular Matrix의 한 예제이다.

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 3 \end{pmatrix} \tag{24}$$

위의 Upper-triangular Matrix (23)는 Linear Systems을 해결함에 있어 Gaussian elimination with back substitute을 적용할 때, 최종 Matrix의 모습임을 알 수 있다.

다시 말해서, 만약에 다음과 같은 Linear Systems가 있다고 해보자.

$$\begin{cases} x + 2y + 3z = 0 \\ 2x + 6y + 9z = 0 \\ 3x + 6y + 12z = 0 \end{cases} \tag{25}$$

위의 식 (25)을 Augmented Matrix로 작성하면 다음과 같다.

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 0 \\ 2 & 6 & 9 & 0 \\ 3 & 6 & 12 & 0 \end{array}\right) \tag{26}$$

위의 행렬 (27)에 Gaussian elimination을 적용하게 되면 다음과 같고,

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 0 \\ 0 & 2 & 3 & 0 \\ 0 & 0 & 3 & 0 \end{array}\right) \tag{27}$$

이는 위에서 살펴본 Upper-triangular Matrix (23)와 동일하고(이 경우에 한해서만), Upper-triangular Matrix 형태임을 확인해볼 수 있다. (중요한 것은 Upper-triangular Matrix 형태라는 사실이다.)

결론적으로, Upper-triangular인 Augmented Matrix (27)을 Linear Systems로 변환하면 다음과 같고,

$$\begin{cases} x + 2y + 3z = 0 \\ 2y + 3z = 0 \\ 3z = 0 \end{cases} \tag{28}$$

마지막으로, Back substitute를 적용하게 되면 다음과 같다. (참고로, Back substitute란 용어는 필자 생각에

Linear systems를 Gaussian elimination을 적용하여 Upper-triagular Matrix를 도출하는 방향은 위에서 아래이지만, Substitute(치환)는 아래에서 위 방향으로 거슬러 올라가기 때문에 Back이란 단어가 접두사로 붙어 Back substitute라 말하는 것으로 이해하였다.)

$$\begin{cases} x + 2y + 3z = 0 \ (x = 0; \ \because y = 0, z = 0) \\ 2y + 3z = 0 \ (y = 0, z = 0; \ \because z = 0) \\ 3z = 0 \ (z = 0) \end{cases} \tag{29}$$

다시 본론으로 돌아와서, Numpy에는 어떤 Square Matrix를 Triangular Matrix로 변환해주는 함수를 제공하는데, **tril**와 **triu** 함수들이 바로 그것이다.

tril 함수는 인자로 전달되는 Square Matrix를 Lower-triangular Matrix로 변환해주고, **triu** 함수는 인자로 전달되는 Square Matrix를 Upper-triangular Matrix로 변환해준다.

다음 예제는 우선 3×3의 Square Matrix를 정의하고, 그것을 Lower-triangular와 Upper-triangular Matrix로 변환하는 과정이 수록되었다.

```
from numpy import array
from numpy import tril
from numpy import triu
# Square Matrix M 정의
M = array([
    [1, 2, 3],
    [1, 2, 3],
    [1, 2, 3]])
print(M)
# Lower-triangular Matrix
lower = tril(M)
print(lower)
# Upper-triangular Matrix
upper = triu(M)
print(upper)
```

B References

[1] Kuldeep Singh. *Linear Algebra: Step by Step*. Oxford University Press, 2013.