IBF Applied Mathematics for Machine Learning Part I: Linear Algebra

Yilin Wu

1 Introduction

Linear algebra is a field of mathematics that is worldwide agreed to be a prerequisite to a better and deeper understanding of machine learning. Although linear algebra is a huge field with many extraordinary findings and theories, some basic tools and notations taken from this field are practical for machine learning practitioners. After completing this section, we will know:

- Linear algebra is the key operation behind data
- Linear algebra has an remarkable impact on statistics
- Linear algebra is highly related to many mathematical tools, such as computer graphics.

1.1 Linear Algebra

Linear algebra is a branch of mathematics. Linear algebra is the mathematics of data, and matrices and vectors are the language of data. It is a relatively young field of study, having initially been formalized in 1800s in order to find unknowns in systems of linear equations.

The application of linear algebra in computer is often called numerical linear algebra. Efficient implementation of vector and matrix operations were originally implemented in the FORTRAN programming language in the 1970s and 1980s, and now calculation were preformed using modern programming languages, such as Python. Three popular open source numerical linear algebra libraries that implement these functions are:

- Linear Algebra Package, or LAPACK
- Basic Linear Algebra Subprograms, or BLAS
- Automatically Tuned Linear Algebra Software, or ATLAS.

1.2 Application of Linear Algebra

Linear algebra are used as a tool in many domains. In one classic book on the topic titled *Introduction to Linear Algebra*, Gilber Strang provided a chapter dedicated to the applications of linear algebra. Briefly they are:

- Computer Graphics, such as the various translation, re-scaling and rotation of images
- Linear Programming, optimization method.
- Graphs and Network, such as analyzing networks.
- Markov Matrices, Population, and Economics, such as population growth

1.3 Examples of Linear Algebra in Machine Learning

In this section, we will review some obvious and concrete examples of linear algebra in machine learning.

1.3.1 Dataset and Data Files

In machine learning, we fit a model for a data set. This is the table like set of numbers where each row represents an observation and each column represents a feature of the observation.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

This data is in fact a **matrix**, a key data structure in linear algebra. Further, when you split the data into inputs and outputs to fit a supervised machine learning model, such as the measurements (matrix) and the survival (vector).

1.3.2 Images and Photographs

If you want to work with images or photographs in computer vision applications, each image that you work with is itself a table structure with a width and height and one pixel value in each cell for black and white images or 3 pixel values in each cell for a color image (RGB). Therefore, a photo is another example of a matrix from linear algebra. Operations on the image, such as cropping, scaling, shearing and so on are all described using the notation and operation of linear algebra.

1.3.3 One Hot Encoding

Sometimes we need to work with categorical data in machine learning. It is common to encode categorical variables to make them easier to work with and learn by the techniques. A popular encoding for categorical data is the **one hot encoding**. A one hot encoding is where a table is created to

represent the variable with one column for each category and a row for each example in the data set.

1.3.4 Linear Regression

Linear regression is an old method from statistics for describing the relationship between variables. It is often used in machine learning for predicting numerical values in simpler regression problem. The most common way of solving linear regression is via a least squares optimization that is solved using matrix factorization method from linear regression, such as an LU decomposition or an singular-value decomposition or SVD.

1.3.5 Principle Component Analysis

Often a data set has many columns, perhaps tens, hundreds, or more. Modeling data with many features is challenging, and models built from data that include irrelevant features are often less skillful than models trained from the most relevant data. It is often hard to know which features of the data are relevant and which are not. Methods for automatically reducing the number of columns of a data set are called dimensionality reduction, and perhaps the most popular one is called principle component analysis or PCA for short.

1.3.6 Singular-Value Decomposition

Another popular dimensionality reduction method is the singular-value-decomposition method or SVD for short. It is a matrix factorization method from the field of linear algebra. It can be directly applied for feature selection, visualization, noise reduction and more.

1.3.7 Latent Semantic Analysis

In the sub-field of machine learning for working with text data called natural language processing, it is common to represent documents as large matrices

of word occurrences. It is a sparse matrix. Matrix factorization methods such as the SVD can be applied to this sparse matrix which has the effect of distilling the representation down to its most relevant essence. This form of data preparation is called LSA or LSI.

1.3.8 Recommender Systems

Predictive modeling problems that involve the recommendation of products are called recommender system. A simple example is in the calculation of hte similarity between sparse customer behavior vectors using distance measures such as Euclidean distance or dot product.

1.3.9 Deep Learning

In deep learning, the execution of neural networks involves linear algebra data structures multiplied and added together. Scaled up to multiple dimensions, deep learning methods work with vectors, matrices and even tensors of inputs and coefficients, where a tensor is a matrix with more than two dimensions.

2 Scalars, vectors, matrices and tensors

Definition: In mathematics, **Scalars** are single numbers.

Definition: In mathematics, **Vectors** are ordered arrays of single numbers.
Example: $[1, 2, 3]$

Exercise: How to create an array?

Definition: In mathematics, **Matrices** are rectangular arrays consisting of numbers.
Example:

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

Exercise: How to create a matrix? (hint: 2d array)

Definition: In mathematics, A **Tensors** is an algebraic object that describes a relationship between sets of algebraic objects related to a vector space.
Example: Scalars are an example of a 0th-order tensor
Vectors are an example of a 1st-order tensor
Matrices are an example of a 2nd-order tensor

Exercise: How to create an tensor?

3 Types of Matrices

There are 6 main types of the matrices; they are:

- Square Matrix
- Symmetric Matrix
- Triangular Matrix
- Diagonal Matrix
- Identity Matrix

3.1 Square Matrix

Definition: A **square matrix** is a matrix where the number of rows (n) is equivalent to the number of columns (n).

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

Square matrices are readily added and multiplied together and are the basis of many simple linear transformations, such as rotations.

3.2 Symmetric Matrix

Definition: A **symmetric matrix** is a type of square matrix where the top-right triangle is the same as the bottom-left triangle.

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

A symmetric matrix is always square and equal to its own **transpose**. The transpose is an operation that flips the number of rows and columns. It is explained in more detail in the next chapter.

$$M = M^T$$

3.3 Triangular Matrix

Definition: A **triangular matrix** is a type of square matrix that has all values in the upper-right or lower-left of the matrix with the remaining elements filled with zero values.

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

Create a triangular matrix with Python

3.4 Diagonal Matrix

Definition: A **diagonal matrix** is one where values outside of the main diagonal have a zero value, where the main diagonal is taken from the top left of the matrix to the bottom right.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 8 \end{bmatrix}$$

Exercise: Create a diagonal matrix from an existing matrix. Transform a

vector into a diagonal matrix

3.5 Identity Matrix

Definition: An **identity matrix** is a square matrix that does not change a vector when multiplied. All of the scalar values along the main diagonal have the value one, while all other values are zero.

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

Exercise: Create an identity matrix with dimension 7.

4 Matrices and Matrix Arithmetic

4.1 Matrix Addition and Subtraction

Two matrices with the same dimensions can be added or subtracted to create a new third matrix. The scalar elements in the resulting matrix are calculated as the addition of the elements in each of the matrices being added. *implementation in Python*

4.2 Matrix Multiplication (Hadamard Product)

Two matrices with the same size can be multiplied together, and this is often called element-wise matrix multiplication or the Hadamard product. It is not the typical operation meant when referring to matrix multiplication. *implementation in Python*

4.3 Matrix Division

One matrix can be divided by another matrix with the same dimensions element-wise. *implementation in Python*

4.4 Matrix-Matrix Multiplication

Matrix multiplication, also called the matrix dot product is more complicated than the previous operations and involves a rule as not all matrices can be multiplied together.

$$C = A \cdot B$$

The rule for matrix multiplication is as follow: The number of columns (n) of the matrix A must equal the number of rows (m) in the second matrix B. If A is of the shape $m \times n$ and B is of the shape $n \times p$, then C is of the shape $m \times p$. The intuition for the matrix multiplication is that we are calculating the dot product between each row in matrix A with each column in matrix B. *implementation in Python*

4.5 Matrix-Vector Multiplication

A matrix and a vector can be multiplied together as long as the rule of matrix multiplication is observed. Specifically, that the number of columns in the matrix must equal the number of items in the vector. *implementation in Python*

4.6 Matrix-Scalar Multiplication

A matrix can be multiplied by a scalar. This can be represented using the dot notation between the matrix and the scalar. We just multiply the number with every element we have in the matrix.

5 Matrix Operation

5.1 Transpose

A defined matrix can be transposed, which creates a new matrix with the number of columns and rows flipped. This is denoted by the transcript T next to the matrix A^T.

$$C = A^T$$

Note: The columns of A^T are the rows of A. *implementation in Python*

5.2 Inverse

Matrix inverse is a process that finds another matrix that when multiplied with the matrix results in an identity matrix. Given a matrix A, find matrix B, such that $AB = I^n$ or $BA = I^n$. The operation of inverting a matrix is indicated by a -1 superscript next to the matrix; for example A^{-1}. The result of the operation is referred to as the inverse of the original matrix; for example, B is the inverse of A.

$$B = A^{-1}$$

Not all matrices are invertible. A square matrix that is not invertible is referred to as singular. *implementation in Python*

5.3 Determinant

The determinant of a square matrix is a scalar representation of the volume of the matrix. It is donated by the $det(A)$ notation or $|A|$, where A is the matrix on which we are calculating the determinant. The determinant of a square matrix is calculated from the elements of the matrix. Moreover, the determinant is the product of all the eigenvalues of the matrix. Eigenvalues are introduced in the lessons on matrix factorization. The intuition for the determinant is that it describes the way a matrix will scale another matrix

when they are multiplied together. For example, the determinant of 1 preserves the space of the other matrix. A determinant of 0 indicates that the matrix cannot be inverted. In NumPy, the determinant of a matrix can be calculated using the `det()` function. *implementation in Python*

5.4 Trace

A trace of a square matrix is the sum of the values on the main diagonal of the matrix (top-left to bottom-right). The operation of calculating a trace on a square matrix is described using the notation $tr(A)$, where A is the square matrix on which the operation is being performed. We can calculate the trace of a matrix in NumPy using the `trace()` function. *implementation in Python*

5.5 Rank

The rank of a matrix is the estimate of the number of linearly independent rows or columns in a matrix. The rank of a matrix M is often denoted as the function `rank()`.

5.5.1 Linear independence

In the theory of vector spaces, a set of vectors is said to be linearly dependent if at least one of the vectors in the set can be defined as a linear combination of the others; if no vector in the set can be written in this way, then the vectors are said to be linearly independent. These concepts are central to the definition of dimension.A vector space can be of finite-dimension or infinite-dimension depending on the number of linearly independent basis vectors. The definition of linear dependence and the ability to determine whether a subset of vectors in a vector space is linearly dependent are central to determining a basis for a vector space.

5.5.2 Geometric Meaning of Ranks

An intuition for rank is to consider it the number of dimensions spanned by all of the vectors within a matrix. For example, a rank of 0 suggest all vectors span a point, a rank of 1 suggests all vectors span a line, a rank of 2 suggests all vectors span a two-dimensional plane. The rank is estimated numerically, often using a matrix decomposition method. NumPy provides the $matrix_{rank}()$ function for calculating the rank of an array.

implementation in Python

6 Vector Norms

Calculating the length or magnitude of vectors is often required either directly as a regularization method in machine learning, or as part of broader vector or matrix operations. The ways to calculate vector lengths or magnitudes, called the vector norm. The length of the vector is always a positive number, except for a vector of all zero values.

6.1 Vector L1 Norm

The length of a vector can be calculated using the L^1 norm, where the 1 is a superscript of the L. The notation for the L^1 norm of a vector is $||v||_1$, where 1 is a subscript. Sometimes, we also call the L^1 norm, the Manhattan norm or taxicab norm. The L^1 norm is calculated as the sum of the absolute vector values, where the absolute value of a scalar uses the notation $|a_1|$.

The L^1 norm of a vector can be calculated in Numpy using the `norm()` function with a parameter to specify the norm order, in this case 1. *implementation in Python*

6.2 Vector L2 Norm

The length of a vector can be calculated using the L^2 norm also.

$$L^2(v) = ||v||_2$$

The L^2 norm calculates the distance of the vector coordinate from the origin of the vector space. As such, it is also known as the Euclidean norm as it is calculated as the Euclidean distance from the origin. The L^2 norm is calculated as the square root of the sum of the squared vector values.

The L^2 norm of a vector can be calculated in NumPy using the `norm()` function with default parameters.
implementation in Python
Note: By far, the L^2 norm is more commonly used than other vector norms in machine learning.

6.3 Vector Max Norm

The length of a vector can be calculated using the maximum norm, also called max norm. Max norm of a vector is referred to as L^{inf} where inf is a superscript and can be represented with the infinity symbol. The notation for max norm is $||v||_{inf}$, where inf is a subscript.

$$L^{inf}(v) = ||v||_{inf}$$

The max norm is calculated as returning the max value of the vector, hence the name. The max norm of a vector can be calculated in NumPy using the `norm()` function with the order parameter set to `inf`.
implementation in Python
Max norm is also used as a regularization in machine learning, such as on neural network weights, called max norm regularization.

7 Systems of linear equations

Definition In mathematics, a **system of linear equations** is a collection of two or more linear equations involving the same set of variables.
Example:

$$4x + 3y = 20$$

$$-5x + 9y = 26$$

There are different ways of solving this problem analytically or numerically. Analytically, we can apply Elimination of Variables, Cramer's Rule, Row Reduction methods, and numerically, we can apply Row Reduction techniques and the Matrix Solution.

7.1 Solving systems of linear equation

In the matrix solution, the system of linear equation to be solved is represented in the form of matrix $AX = B$. For instance, we can represent equation from the previous example as follows

$$A = \begin{bmatrix} 4 & 3 \\ -5 & 9 \end{bmatrix}$$

$$X = \begin{bmatrix} x \\ y \end{bmatrix} B = \begin{bmatrix} 20 \\ 26 \end{bmatrix}$$

To find the value of x and y variables, we need to find the values in the matrix X. To do so, we can take the dot product of the inverse of matrix A, and the matrix B as follows:

$$X = A^{-1}B$$

implementation in Python
Exercise

$$4x + 3y + 2z = 25$$

$$-2x + 2y + 3z = -10$$

$$3x - 5y + 2z = -4$$

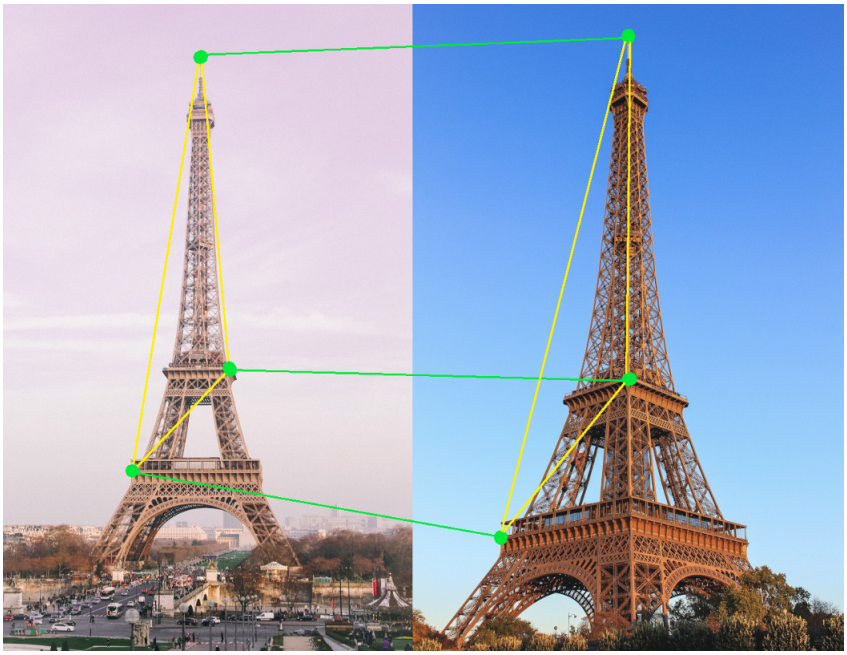

Figure 1: Find the transformation between the image of the Eiffel Tower, photo credit: Jungxon Park

Suppose, a fruit seller sold 20 mangoes and 10 oranges in one day for a total of 350. The next day he sold 17 mangoes and 22 orange for 500. If the prices of the fruits remained unchanged on both the days, what was the price of one mango and one orange?

8 Linear transformation

Linear transformation (linear map, linear mapping or linear function) is a mapping $V \to W$ between two vector spaces, that preserves addition and scalar multiplication. Linear transformations are often used in machine learning applications. They are useful in the modeling of 2D and 3D animation, where an objects size and shape needs to be transformed from one viewing angle to the next. An object can be rotated and scaled within a space using

a type of linear transformations known as geometric transformations, as well as applying transformation matrices.

8.1 Transformation matrix

8.1.1 Stretching

A stretch in the xy-plane is a linear transformation which enlarges all distances in a particular direction by a constant factor but does not affect distances in the perpendicular direction. We only consider stretches along the x-axis and y-axis.

Note that if k is ¿ 1, then this really is a "stretch"; if k is ¡ 1, it is technically a "compression", but we still call it a stretch. Also, if k=1, then the transformation is an identity, i.e. it has no effect. The matrix associated with a stretch by a factor k along the x-axis is given by

$$\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}$$

The matrix associated with a stretch by a factor k along the y-axis is given by

$$\begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$$

8.1.2 Rotation

Rotation by an angle θ clockwise about the origin is given by

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Rotation by an angle θ counterclockwise about the origin is given by

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

8.1.3 Shearing

A shear parallel to the x-axis is given by

$$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$$

A shear parallel to the y-axis is given by

$$\begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix}$$

8.1.4 Reflection

For reflection about a line that goes through the origin, let $l = (l_x, l_y)$ be a vector in the direction of the line. Then use the transformation matrix:

$$A = \frac{1}{||l||^2} \begin{bmatrix} 1_x^2 - l_y^2 & 2l_x l_y \\ 2l_x l_y & l_y^2 - l_x^2 \end{bmatrix}$$

Implementation in Python

9 Factorization

9.1 Matrix Decomposition

Many complex matrix operations cannot be solved efficiently or with stability using the limited precision of computers. Matrix decompositions are methods that reduce a matrix into constituent parts that make it easier to calculate more complex matrix operations. Matrix decomposition methods, also called matrix factorization methods, are a foundation of linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.

9.1.1 LU Decomposition

The LU decomposition is for square matrices and decomposes a matrix into L and U components.

$$A = LU$$

Where A is the square matrix that we wish to decompose, L is the lower triangle matrix and U is the upper triangle matrix.

The LU decomposition can be implemented in Python with the `lu()` function. More specifically, this function calculates the LU composition with partial pivoting.

Implementation in Python

9.1.2 QR Decomposition

The QR decomposition is for $n \times m$ matrices (not limited to square matrices) and decomposes a matrix into Q and R components.

$$A = QR$$

Where A is the matrix that we wish to decompose, Q a matrix with the size $m \times m$, and R is an upper triangle matrix with the size $m \times n$.

The QR decomposition can be implemented in NumPy using the `qr()` function. By default, the function returns the Q and R matrices with smaller or reduced dimensions that is more economical. We can change this to return the expected sizes of $m \times m$ for Q and $m \times n$ for R by specifying the mode argument as `complete`.

Implementation in Python

9.1.3 Cholesky Decomposition

The Cholesky decomposition is for square symmetric matrices where all values are greater than zero, so-called positive defi

nite matrices. For our interests in machine learning, we will focus on the Cholesky decomposition for real-valued matrices and ignore the cases when working with complex numbers. The decomposition is defi

ned as follows:

$$A = LL^T$$

Where A is the matrix being decomposed, L is the lower triangular matrix and L^T is the transpose of L.

The Cholesky decomposition can be implemented in `NumPy` by calling the `cholesky()` function.
Implementation in Python

9.2 Eigenvalue decomposition

The most used type of matrix decomposition is the eigen-decomposition that decomposes a matrix into eigen-vectors and eigenvalues. This decomposition also plays a role in methods used in machine learning, such as in the **Principal Component Analysis** method or PCA.

A vector is an eigen-vector of a matrix if it satisfi

es the following equation.

$$Av = \lambda v$$

This is called the eigenvalue equation, where A is the parent square matrix that we are decomposing, v is the eigenvector of the matrix, and λ is the lowercase Greek letter lambda and represents the eigenvalue scalar. A matrix could have one eigenvector and eigenvalue for each dimension of the parent matrix. Not all square matrices can be decomposed into eigen-vectors and eigenvalues, and some can only be decomposed in a way that requires complex numbers. The parent matrix can be shown to be a product of the eigenvectors and eigenvalues.

$$A = Q\lambda A^T$$

Where Q is a matrix comprised of the eigenvectors, Λ is the uppercase Greek letter lambda and is the diagonal matrix comprised of the eigenvalues, and

Q^T is the transpose of the matrix comprised of the eigenvectors.
Implementation in Python

9.2.1 Reconstruct Matrix

We can reverse the process and reconstruct the original matrix given only the eigenvectors and eigenvalues. First, the list of eigenvectors must be taken together as a matrix, where each vector becomes a row. The eigenvalues need to be arranged into a diagonal matrix. The `NumPy diag()` function can be used for this. Next, we need to calculate the inverse of the eigenvector matrix, which we can achieve with the `inv()` function. Finally, these elements need to be multiplied together with the `dot()` function.
Implementation in Python

9.3 Singular value decomposition

Perhaps the most known and widely used matrix decomposition method is the Singular-Value Decomposition, or SVD. All matrices have an SVD, which makes it more stable than other methods, such as the eigen-decomposition. As such, it is often used in a wide array of applications including compressing, denoising, and data reduction.
The Singular-Value Decomposition, or SVD for short, is defined by:

$$A = U\Sigma V^T$$

Where A is the real $n \times m$ matrix that we wish to decompose, U is an $m \times m$ matrix, Σ represented by the uppercase Greek letter sigma) is an $m \times n$ diagonal matrix, and V^T is the transpose of an $n \times n$ matrix where T is a superscript. The diagonal values in the Σ matrix are known as the singular values of the original matrix A. The columns of the U matrix are called the left-singular vectors of A, and the columns of V are called the right-singular vectors of A.
Implementation in Python

9.3.1 Reconstruct Matrix

Same idea as eigen-decomposition, we are able to reconstruct the matrix by doing the reverse operation.
Implementation in Python

9.3.2 The Moore-Penrose pseudo inverse

The pseudo inverse is the generalization of the matrix inverse for square matrices to rectangular matrices where the number of rows and columns are not equal. It is also called the Moore-Penrose Inverse after two independent discoverers of the method or the Generalized Inverse. The pseudoinverse is denoted as A^+, where A is the matrix that is being inverted and + is a superscript. The pseudoinverse is calculated using the singular value decomposition of A:

$$A^+ = VD^+U^T$$

Where A^+ is the pseudoinverse, D^+ is the pseudoinverse of the diagonal matrix Σ and V^T is the transpose of V . We can get U and V from the SVD operation.

$$A = U\Sigma V^T$$

Implementation in Python

9.3.3 Dimension Reduction

A popular application of SVD is for dimensionality reduction. Data with a large number of features, such as more features (columns) than observations (rows) may be reduced to a smaller subset of features that are most relevant to the prediction problem. The result is a matrix with a lower rank that is said to approximate the original matrix. To do this we can perform an SVD operation on the original data and select the top k largest singular values in Σ. These columns can be selected from Σ and the rows selected from V^T . An approximate B of the original vector A can then be reconstructed.

$$B = U\Sigma_k V_k^T$$

In natural language processing, this approach can be used on matrices of word occurrences or word frequencies in documents and is called Latent Semantic Analysis or Latent Semantic Indexing. In practice, we can retain and work with a descriptive subset of the data called T. This is a dense summary of the matrix or a projection.

$$T = U\Sigma_k$$

Implementation in Python

The scikit-learn provides a `TruncatedSVD` class that implements this capability directly.

Implementation in Python