Vectores y Matrices

Anexos de Vectores y Matrices

Los Vectores y su Aritmética

Los elementos fundamentales del **Álgebra Lineal** son **los vectores.** Los vectores son ampliamente utilizados **en toda el área de Machine Learning** para la **descripción de algoritmos** para el entrenamiento de modelos.

¿Qué es un Vector?

Un vector es **una tupla de uno o más elementos llamados escalares.** Éstos se construyen a partir de componentes, que son números ordinarios. Se puede pensar en el vector como **una lista de números** utilizadas para realizar operaciones.

Los vectores se representan usualmente **con una letra minúscula.** Por ejemplo:

$$\mathrm{v} = (\mathrm{v}_1, \mathrm{v}_2, \mathrm{v}_3)$$

En donde cada elemento es un valor escalar, usualmente valores reales. Además, se pueden representar de manera vertical, por ejemplo:

$$\mathrm{v} = \begin{pmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \\ \mathrm{v}_3 \end{pmatrix}$$

Es común representar **la variable objetivo como un vector de nombre** y, al momento de entrenar un modelo de Machine Learning. También es común representar a un vector **con una analogía geométrica,** donde cada entrada del vector es una coordenada en un **espacio n-dimensional.**

Las analogías son un buen punto de partida, pero no se deben tomar tan en serio, pues la perspectiva **se pierde una vez que se manejan altas dimensionalidades.**

Definiendo un Vector en Python

En Python, **se puede representar un vector en un arreglo de NumPy.** Por ejemplo, en el *ANEXO 1.*

Aritmética de un Vector

En esta parte veremos **cómo funciona la aritmética de un vector.** Todas las operaciones se realizan elemento por elemento entre dos vectores. Veamos cuáles son **las operaciones principales**.

Suma de Vectores

En cuanto a la suma, **dos vectores pueden ser sumados si tienen la misma longitud.** Al sumarlos se forma un tercer vector **de la misma longitud que los vectores iniciales.**

$$c = a + b$$

Cada elemento del nuevo vector **se calcula sumando los dos elementos del mismo índice en los vectores iniciales.** Por ejemplo:

$$c = (a_1 + b_1,\ a_2 + b_2,\ a_3 + b_3)$$

En Python podemos realizar esta operación tal y como se muestra en el *ANEXO 2.*

Resta de Vectores

Es posible restarle un vector a otro vector de la misma longitud **creando otro vector.**

$$c = a - b$$

Al igual que en la suma **la resta se hace elemento a elemento.** Por ejemplo:

$$c = (a_1 - b_1,\ a_2 - b_2,\ a_3 - b_3)$$

En Python podemos realizar esta operación tal y como se muestra en el *ANEXO 2.*

Multiplicación Vectorial

Dos vectores de la **misma longitud pueden ser multiplicados**:

$$c = a \times b$$

Al igual que las operaciones que vimos **la multiplicación se realiza entrada a entrada,** resultando en un nuevo vector con la misma longitud.

$$c = (a_1 \times b_1, a_2 \times b_2, a_3 \times b_3)$$

En Python podemos realizar esta operación tal y como se muestra en el *ANEXO 2.*

División de Vectores

Dos vectores de la misma longitud se pueden dividir y la notación es la siguiente:

$$c = \frac{a}{b}$$

Como en las otras operaciones, **la división se realizará elemento a elemento,** creando así un nuevo vector de la misma longitud.

$$c = (\frac{a_1}{b_1}, \frac{a_2}{b_2}, \frac{a_3}{b_3})$$

En Python podemos realizar esta operación tal y como se muestra en el *ANEXO 2.*

Producto Punto Vectorial

El producto punto se calcula como **la suma de la multiplicación de los elementos entre dos vectores del mismo tamaño.** Esto da como resultado un **escalar.** Esta operación recibe este nombre por la notación que se utiliza como símbolo.

El producto punto es **una herramienta de gran utilidad** para calcular **proyecciones vectoriales, descomposiciones vectoriales y para determinar la ortogonalidad.**

Esta operación puede ser utilizada en **Machine Learning** para calcular la suma ponderada de un vector. En términos de vectores, se calcula el producto punto como:

$$c = (a_1 \times b_1 + a_2 \times b_2 + a_3 \times b_3$$

En Python, se calcula el producto punto utilizando la función **dot()** que se escribe como se muestra en el *ANEXO 2.*

Multiplicación por un Escalar

Un vector puede ser multiplicado por un escalar, que tiene como efecto **el cambio de magnitud del vector.** Para no confundirse con la notación, se utilizará s para describir al valor escalar, de forma que la multiplicación escalar se verá como:

$$c = s \times v$$

La multiplicación escalar se **realiza a cada elemento del vector**, un ejemplo de esto sería:

$$c = (s \times v_1, s \times v_2, s \times v_3)$$

Finalmente, se muestra cómo se realiza esta operación en el *ANEXO 2.*

Norma Vectorial

Calcular la **longitud o magnitud** de un vector es muy usado como herramienta para **el método de regularización en Machine Learning.** Ahora toca ver diferentes maneras de calcular las longitudes y magnitudes de un vector. Esto recibe el nombre de **Norma de Vector.**

Norma de un Vector

La norma de un vector es un número no negativo que describe la **extensión del vector en el espacio** y a veces se le conoce como la **magnitud del vector.** Se calcula utilizando una medida que resume **la distancia del origen al vector.** A continuación, veremos algunas maneras comunes de calcular la norma.

Norma L^1

La magnitud de un vector se puede calcular utilizando la norma L^1, en donde el **1** es el superíndice de **L.** La notación de la norma es $||v||_1$ donde **1** es el subíndice. También se le conoce a esta norma como la **norma del taxista.** Y se escribe como $L^1(v) = ||v||_1$.

Esta norma se calcula como **la suma de los valores absolutos de cada elemento.**

$$||v_1|| = |a_1| + |a_2| + |a_3|$$

En algunas aplicaciones de **Machine Learning** es importante discriminar entre los elementos que son exactamente cero y los elementos que son cercanos a cero. En esos casos, **es útil utilizar la norma L^1,** pues mantiene constante la tasa de crecimiento.

En Python se puede sacar la norma de un vector utilizando la función $norm()$ en donde se utiliza el segundo parámetro para especificar el **orden de la norma**. El ejemplo de esto se encuentra en el *ANEXO 3.*

Norma L^2

La notación para esta norma es la siguiente: $\mathrm{L}^2(\mathrm{v}) = ||\mathrm{v}||_2$

Se calcula como **la raíz cuadrada del cuadrado de los elementos del vector:**

$$||\mathrm{v}||_2 = \sqrt{\mathrm{a}_1^2 + \mathrm{a}_2^2 + \mathrm{a}_3^2}$$

Para calcularlo en Python se utiliza como parámetro al **2**. El ejemplo se encuentra en el *ANEXO 3.*

Este método tiene como ventaja el que **deja que los coeficientes sean pequeños por la raíz cuadrada** y esto hace que el modelo sea menos complejo.

Norma Max

La norma máxima de un vector se le puso como nombre L^{inf} donde el inf se puede también representar por el infinito: $\mathrm{L}^\infty(\mathrm{v}) = ||\mathrm{v}||_\infty$

Se calcula **obteniendo el máximo elemento en su valor absoluto.**

$$||\mathrm{v}||_\infty = \max\{|\mathrm{v}_1|, |\mathrm{v}_2|, |\mathrm{v}_3|\}$$

Para calcularlo en Python se utiliza como parámetro al **inf**. El ejemplo se encuentra en el *ANEXO 3.*

Las Matrices y su Aritmética

Las matrices son un **elemento fundamental** en Álgebra Lineal. Se utilizan extensivamente en Machine Learning en la **descripción de algoritmos y procesos** como las variables de entrada (X).

¿Qué es una Matriz?

La matriz **es un arreglo multiescalar** con una o más columnas y una o más renglones.

La notación para una matriz es **usualmente una letra mayúscula** como A y las entradas son localizadas por sus subíndices, renglones **(i)** y sus columnas **(j)** como por ejemplo, a_{ij}. Por ejemplo, veamos una matriz de 3×2.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix}$$

Un lugar muy común en donde se encontrarán estas matrices será en **el entrenamiento de modelos para guardar la información.** Una matriz se define en Python como **un arreglo bidimensional**. El ejemplo de esto se encuentra en el *ANEXO 4.*

Aritmética de las Matrices

A continuación, vamos a definir cómo se realizarán las operaciones aritméticas como suma, resta y multiplicación.

Adición Matricial

La adición matricial es muy simple y funciona como en los vectores, es decir sea realiza la suma elemento a elemento de las matrices. Para esto se necesita que las matrices sean del mismo tamaño. De manera que si lo que queremos hacer es: $C = A + B$, se ve de la siguiente manera.

$$C = \begin{pmatrix} a_{1,1} + b_{1,1} & a_{1,1} + b_{1,1} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} \\ a_{3,1} + b_{3,1} & a_{3,2} + b_{3,2} \end{pmatrix}$$

Un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

Resta Matricial

La resta matricial funciona de la misma manera que la suma, sólo que, en este caso el orden importa, así como en los números reales. Así si buscamos hacer es: $C = A - B$, se realiza de la siguiente manera:

$$C = \begin{pmatrix} a_{1,1} - b_{1,1} & a_{1,1} - b_{1,1} \\ a_{2,1} - b_{2,1} & a_{2,2} - b_{2,2} \\ a_{3,1} - b_{3,1} & a_{3,2} - b_{3,2} \end{pmatrix}$$

Nuevamente, un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

Producto Hadamard de Matrices

El producto Hadamard, no sigue las reglas usuales de multiplicación de matrices, por lo cual se utiliza un operador distinto para diferenciarlo. Esta multiplicación se hace multiplicando elemento a elemento tal como la suma y la resta. De manera que si se busca hacer: $C = A \circ B$, se realiza de la siguiente manera:

$$C = \begin{pmatrix} a_{1,1} \times b_{1,1} & a_{1,1} \times b_{1,1} \\ a_{2,1} \times b_{2,1} & a_{2,2} \times b_{2,2} \\ a_{3,1} \times b_{3,1} & a_{3,2} \times b_{3,2} \end{pmatrix}$$

Nuevamente, un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

División Matricial

La división matricial se realiza de igual manera que el Producto Hadamard, pues en el sentido matemático, no se define una división matricial. De manera que si se busca hacer $C = \frac{A}{B}$, se realiza de la siguiente manera:

$$C = \begin{pmatrix} \frac{a_{1,1}}{b_{1,1}} & \frac{a_{1,1}}{b_{1,1}} \\ \frac{a_{2,1}}{b_{2,1}} & \frac{a_{2,2}}{b_{2,2}} \\ \frac{a_{3,1}}{b_{3,1}} & \frac{a_{3,2}}{b_{3,2}} \end{pmatrix}$$

Nuevamente, un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

Multiplicación Matricial

Ahora sí definiremos cómo se hace la multiplicación matricial como se acostumbra en matemáticas. Para realizar esta multiplicación se tienen que cumplir diferentes criterios a los anteriores, ahora necesitaremos que el número de columnas del primer multiplicando sea igual al número de renglones de segundo multiplicando. Realizar la operación resultará en una matriz con el número de renglones del primer multiplicando y el número de columnas del segundo multiplicando.

La intuición sobre la multiplicación matricial se basa en que estamos realizando un producto punto entre cada renglón del primer multiplicando con las columnas del segundo multiplicando. Digamos que se quiere realizar la operación $C = A \cdot B = AB$, y se tiene que las matrices son de la forma:

$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix}$ y $B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$ al ser multiplicadas resulta en lo siguiente:

$$C = \begin{pmatrix} a_{1,1} \times b_{1,1} + a_{1,2} \times b_{2,1} & a_{1,1} \times b_{1,2} + a_{1,2} \times b_{2,2} \\ a_{2,1} \times b_{1,1} + a_{2,2} \times b_{2,1} & a_{2,1} \times b_{1,2} + a_{2,2} \times b_{2,2} \\ a_{3,1} \times b_{1,1} + a_{3,2} \times b_{2,1} & a_{3,1} \times b_{1,2} + a_{3,2} \times b_{2,2} \end{pmatrix}$$

Nuevamente, un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

Multiplicación Matriz-Vector

La multiplicación Matriz-Vector se realiza pensando que un vector de n entradas, se piensa como una matriz de $n \times 1$ entradas. Al hacer esta conversión, se procede a realizar la multiplicación matricial que acabamos de revisar. Nuevamente, un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

Multiplicación Matriz-Escalar

Esta operación sirve para multiplicar todos los elementos de la matriz por el mismo escalar. Esto funciona como si realizáramos una matriz donde todos los elementos son el escalar y hacemos una multiplicación Hadamard de matrices. Nuevamente, un ejemplo de cómo se realiza esto en Python, se encuentra en el *ANEXO 4.*

Tipos de Matrices

En esta sección vamos a ver seis tipos de matrices que cumplen ciertas características muy útiles para realizar operaciones.

Matriz Cuadrada

Una matriz cuadrada es aquella que cumple que el número de renglones es igual al número de matrices, es decir: $M_{n \times n}, \forall n \epsilon \mathbb{N}$ es una matriz cuadrada.

Matriz Simétrica

Una matriz simétrica cumple que el triángulo superior-derecho es igual al triángulo inferior-izquierdo, esto en términos matriciales se escribe $M = M^T$ y cumple que para cada elemento $\mathrm{m_{a,b}} = \mathrm{m_{b,a}}$.

Matriz Triangular

Una matriz triangular es una matriz que tiene todos los elementos del triángulo (superior o inferior) y el resto de los elementos son igual a cero. De manera que una matriz con valores en el triángulo inferior, se le llama matriz triangular inferior y para una matriz que tiene valores en el triángulo superior, se le llama matriz triangular superior.

NumPy permite hacer matrices triangulares provenientes de matrices cuadradas. La muestra de cómo se hace se encuentra en el *ANEXO 5.*

Matriz Diagonal

Las matrices diagonales, denotadas usualmente como D, son aquellas que cumplen que todos los elementos fuera de la diagonal son iguales a cero. Matemáticamente se escribe una matriz diagonal cumple $\mathrm{a} \neq \mathrm{b} \Rightarrow \mathrm{D_{a,b}} = 0$.

NumPy nos proporciona la función *diag()* que crea una matriz diagonal de una matriz cuadrada. El ejemplo se encuentra en el *ANEXO 5*.

Matriz Identidad

Una matriz identidad es una matriz diagonal, pero cada elemento en la diagonal es igual a uno. Lo especial de esta matriz es que, al ser multiplicada por cualquier matriz, resulta la misma matriz. Usualmente se denotan estas matrices como $\mathrm{I^n}$ donde n representa la dimensión de la matriz.

NumPy puede realizar matrices identidad con la función *identity()*, la demostración de su uso se encuentra en el *ANEXO 5*.

Matriz Ortogonal

Dos vectores se les llaman ortogonales si cumplen que su producto punto es igual a cero. Si la norma de ambos vectores es igual a uno, se les llama ortonormales, pues son ortogonales y están normalizados. Esto matemáticamente significa que dos vectores ortogonales cumplen $\mathrm{v} \cdot \mathrm{w}^{\mathrm{T}} = 0$. Y ortonormales si ambos cumplen que $||\mathrm{v}|| = 1$. Ahora que entendemos esta parte, decimos que una matriz cuadrada es ortogonal si sus columnas y renglones son mutuamente vectores ortonormales. Usualmente se les describe con el nombre Q. Lo interesante es que, cumplen que $\mathrm{Q}^{\mathrm{T}} \cdot \mathrm{Q} = \mathrm{Q} \cdot \mathrm{Q}^{\mathrm{T}} = \mathrm{I}$. Es decir, siempre ocurre que $\mathrm{Q}^{\mathrm{T}} = \mathrm{Q}^{-1}$. Un ejemplo de una matriz ortonormal es:

$$\mathrm{Q} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

El ejemplo de las operaciones en cuanto a sus características se encuentra en el *ANEXO 5.*

Operaciones Matriciales

Las operaciones matriciales se utilizan frecuentemente en la descripción de algoritmos de Machine Learning. Pueden resolver rápidamente ecuaciones o servir como herramienta para realizar operaciones matriciales complejas.

Transpuesta

Una matriz puede ser transpuesta que crea una matriz de $\mathrm{n} \times \mathrm{m}$ si la matriz era de $\mathrm{m} \times \mathrm{n}$. Una matriz transpuesta se describe como $\mathrm{C} = \mathrm{A}^{\mathrm{T}}$ y lo que hace elemento a elemento es que convierte $\mathrm{a}_{\mathrm{r,l}}$ en el elemento $\mathrm{a}_{\mathrm{l,r}}$. Cabe mencionar que, si la matriz es simétrica, esta operación no realiza ningún cambio.

Inversa

La inversa de una matriz es aquella que, al multiplicarse por la matriz inicial, resulta la matriz identidad. Es decir, si la matriz B es la inversa de la matriz A se cumple que $\mathrm{AB} = \mathrm{BA} = \mathrm{I}$. La operación de la inversión de una matriz se denota con un superíndice, es decir $\mathrm{B} = \mathrm{A}^{-1}$. El cálculo de la inversa no se computa de manera directa, pero se descubre a través de operaciones numéricas que se verán en secciones posteriores.

Traza

La traza de una matriz cuadrada es la suma de los elementos de la diagonal de la misma matriz. La notación que se utiliza para esta operación es $\mathrm{tr}(\mathrm{A})$ en donde A es la matriz de la que se obtiene la traza. El ejemplo de esta operación se encuentra en el *ANEXO 6*.

Determinante

La determinante de una matriz cuadrada es la representación escalar del volumen de la matriz. Se denota por $\det(\mathrm{A})$ o por $|\mathrm{A}|$ donde A es la matriz a la que se le aplica la determinante. Obtendremos su determinante a partir de sus elementos. De manera más técnica, el determinante es el producto de todos los eigenvalues de la matriz. Sin embargo, se explicarán los eigenvalues en una lección posterior.

La intuición de la determinante es que describe la escala de una matriz en caso de que multiplique por otra matriz y tener una idea de la escala de la matriz resultante. Por ejemplo, si el determinante es igual a uno, entonces la matriz no cambiará la escala, por otro lado, si el determinante es igual a cero, la matriz no podrá ser invertida. Veremos cómo se saca la determinante en Python en el *ANEXO 6*.

Rango

El rango de una matriz es el estimado del número de renglones o columnas linealmente independientes. El rango de la matriz M se denota como $rango(M)$.

La intuición del rango es que considera el número de dimensiones que generan todos los vectores dentro de una matriz. Por ejemplo, un rango de cero dice que los vectores generan un punto, un rango de uno dice que los vectores generan una recta y un rango de dos sugiere que los vectores generan un plano bidimensional. El rango se calcula numéricamente utilizando usualmente el método de la descomposición matricial. Usualmente se utiliza el método *Singular-Value Decomposition* conocida como *SVD* por sus siglas. El ejemplo de cómo se usa en Python se encuentra en el *ANEXO 6*.

Matrices Escasas

Las matrices que contienen en su mayoría ceros se les llaman matrices escasas. Mientras que, al contrario, si una matriz casi no contiene ceros, se le llama matriz densa. Es común manejar matrices escasas grandes y especialmente en el área de Machine Learning. Es computacionalmente costoso trabajar con matrices

escasas como si fuesen densas y se ve útil el uso de representaciones y operaciones que manejan específicamente la escasez de la matriz.

La escasez de una matriz se mide a través de la siguiente fórmula:

$$\text{escasez} = \frac{\#\text{ de ceros}}{\#\text{ de elementos}}$$

Problemas con la escasez

Las matrices escasas pueden ocasionar problemas en cuanto a la complejidad del tiempo y del espacio.

Complejidad Espacial

Matrices muy largas requieren de mucha memoria y algunas de las matrices con las que trabajaremos serán escasas. Un ejemplo de una matriz escasa es cuando se cuentan el número de palabras de un libro. Claramente trabajar estas matrices como matrices densas sería una pérdida de memoria.

Complejidad del Tiempo

Si asumimos que una matriz grande y escasa entra en la memoria, querremos hacer operaciones con la matriz. Sin embargo, si la matriz tiene casi todos sus elementos igual a cero, entonces realizar operaciones tomaría mucho más tiempo al tener que sumar o multiplicar valores igual a cero.

El problema se vuelve peor cuando consideramos que los modelos de Machine Learning utilizan una gran cantidad de operaciones. Lo que perjudicará en gran cantidad el tiempo de procesamiento.

Matrices Escasas en Machine Learning

Datos

Las matrices escasas se presentan en tipos de datos específicos, especialmente en las observaciones de una ocurrencia o una actividad. Los ejemplos son:

- Si un usuario ha visto una película o no de un catálogo de películas.
- Si un usuario ha comprado o no un producto de un catálogo de productos.
- La cuenta de cuántas un usuario ha escuchado una canción de un catálogo de películas.

Preparación de Datos

Las matrices escasas aparecen en esquemas de código para la preparación de datos. Tres ejemplos comunes son:

- One Hot Encoding, usado para representar datos categóricos como vectores escasos binarios.
- Count Encoding, usado para representar la frecuencia de palabras utilizadas en un vocabulario para un documento.
- TF-IDF Encoding, usado para representar la frecuencia normalizada de una palabra en un vocabulario.

Áreas de Estudio

Algunas áreas de estudio dentro de Machine Learning deben desarrollar métodos especializados para trabajar con matrices escasas dado de los datos de entrada casi siempre son escasos. Tres ejemplos serían:

- Procesamiento de Lenguaje Natural para trabajar con documentos de texto.
- Sistemas de recomendación para trabajar con un producto en un catálogo.
- Visión Computacional cuando se trabajan en imágenes con muchos pixeles negros.

Trabajando con Matrices Escasas

La solución para trabajar y representar matrices escasas es utilizar una estructura alterna para representar datos escasos. De manera que se ignoren los datos nulos y sólo se trabaje con los datos distintos de cero.

Existen múltiples estructuras de datos que pueden utilizarse para construir de manera eficiente una matriz escasa. Tres ejemplos comunes de esto son:

- Diccionario de Llaves. Un diccionario se utiliza cuando el índice de cada renglón y columna mapea a un valor.
- Lista de Listas. Cada renglón de la matriz de guarda en una lista y cada sublista contiene el índice de la columna y el valor.
- Lista de Coordenadas. Una lista de tuplas, donde cada tupla contiene los índices del renglón y la columna y el valor de la matriz en dichas coordenadas.

También hay estructuras de datos que le quedan mejor para realizar operaciones eficientes, dos usualmente utilizadas son:

- Renglón Escaso Comprimido. La matriz escasa se representa utilizando tres arreglos unidimensionales para valores no igual a cero, para el grado de los renglones y los índices de las columnas.
- Columna Escasa Comprimida. Igual que la anterior, pero los índices de las columnas se comprimen y se leen antes que los índices de los renglones.

Una muestra de cómo funciona lo anterior se encuentra en el *ANEXO 7*.

Tensores y Aritmética de Tensores

Qué son los Tensores

Un tensor es la generalización de vectores y matrices y se entiende fácilmente como un arreglo multidimensional.

Un vector es un tensor de primer orden y una matriz es un tensor de segundo orden. La notación tensorial es muy parecida a la notación matricial con la letra mayúscula siendo el nombre del tensor y las letras minúsculas se utilizan para representar elementos del tensor. Por ejemplo, pensemos que se tiene el tensor T y cada elemento se denota como $t_{i,j,k}$ donde k sería el valor de la tercera dimensión.

La forma para declarar tensores en Python se muestra en el *ANEXO 8*.

Aritmética de Tensores

Adición Tensorial

La suma de dos tensores se hace elemento a elemento, tal como la suma matricial.

Producto Hadamard Tensorial

El producto Hadamard tensorial se realiza elemento a elemento, tal como se hacía el producto Hadamard de matrices.

División Tensorial

La división tensorial se hace elemento a elemento, tal como se hacía la división matricial.

Producto Tensorial

El operador del producto tensorial es $\otimes$. Dado un tensor A con q dimensiones y un tensor B con r dimensiones, el producto de estos tensores generará un tensor de $q + r$ dimensiones. Es más sencillo explicar cómo se realiza el producto tensorial con un ejemplo. Pensemos que se tienen los siguientes tensores de segundo orden:

$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}, B = \begin{pmatrix} \mathrm{b}_{1,1} & \mathrm{b}_{1,2} \\ \mathrm{b}_{2,1} & \mathrm{b}_{2,2} \end{pmatrix}$ entonces el tensor $C = A \otimes B$ se obtiene así:

$$C = \begin{pmatrix} a_{1,1} \times \begin{pmatrix} \mathrm{b}_{1,1} & \mathrm{b}_{1,2} \\ \mathrm{b}_{2,1} & \mathrm{b}_{2,2} \end{pmatrix} & a_{1,2} \times \begin{pmatrix} \mathrm{b}_{1,1} & \mathrm{b}_{1,2} \\ \mathrm{b}_{2,1} & \mathrm{b}_{2,2} \end{pmatrix} \\ a_{2,1} \times \begin{pmatrix} \mathrm{b}_{1,1} & \mathrm{b}_{1,2} \\ \mathrm{b}_{2,1} & \mathrm{b}_{2,2} \end{pmatrix} & a_{2,2} \times \begin{pmatrix} \mathrm{b}_{1,1} & \mathrm{b}_{1,2} \\ \mathrm{b}_{2,1} & \mathrm{b}_{2,2} \end{pmatrix} \end{pmatrix}$$

$$C = \begin{pmatrix} a_{1,1} \times b_{1,1} & a_{1,1} \times b_{1,2} & a_{1,2} \times b_{1,1} & a_{1,2} \times b_{1,2} \\ a_{1,1} \times b_{2,1} & a_{1,1} \times b_{2,2} & a_{1,2} \times b_{2,1} & a_{1,2} \times b_{2,2} \\ a_{2,1} \times b_{1,1} & a_{2,1} \times b_{1,2} & a_{2,2} \times b_{1,1} & a_{2,2} \times b_{1,2} \\ a_{2,1} \times b_{2,1} & a_{2,1} \times b_{2,2} & a_{2,2} \times b_{2,1} & a_{2,2} \times b_{2,2} \end{pmatrix}$$

La muestra de cómo se realizan todas las operaciones anteriormente mencionadas se encuentran en el *ANEXO 8*.