

# Analytic Network Process

## *Una Estimación del Mejor Proveedor para Natural Response*

*Autor:*

Victor Gonzalez Rodriguez  
victor.gonzalezr@usm.cl

*Profesor:*

Lautaro Guerra  
lautaro.guerra@usm.cl

24 de Junio de 2012

## 1. Benchmark de multiplicaciones

- **Software:** Para desarrollar el programa que realiza el benchmark comparativo, se ocupó la librería *numpy*, para el manejo mejorado de matrices y arrays. Cada una de las funciones de multiplicación, ejecuta un proceso atómico de multiplicaciones para así comparar de mejor manera cuanto demora cada método.
- **Conclusiones:** A partir de las comparaciones hechas entre los distintos métodos usando las matrices propuestas, se ha logrado concluir que el método más eficiente para calcular la multiplicación entre 2 matrices, es el de enfoque fila, seguido por el enfoque columna, y finalmente el producto punto, siendo notablemente más lento que los otros métodos.

En el siguiente resultado de ejemplo, el producto punto llega a ser 20 veces mas lento que el enfoque fila:

Producto punto: 1.66762685776 segundos.

Enfoque Columna: 0.0907769203186 segundos.

Enfoque Fila: 0.0868830680847 segundos.

## 2. Tipos de matriz

- **Software:** Para el desarrollo del software que analiza que tipo de matriz es una matriz A, se utilizó principalmente la librería *numpy* y sus habilidades de computo de álgebra lineal *numpy.linalg* (para calcular la inversa, la conjugada de una matriz, etc.).

- **Conclusiones:** Fue interesante descubrir las propiedades que cumplen los tipos de matrices y como ayudan a determinar ciertos comportamientos y propiedades que pueden ayudar a simplificar el cálculo.

### 3. Transformación de imágenes

- **Software:** Para este software se usó la librería *numpy*, *scipy* y *pylab*. Las funciones generan matrices (imágenes), las cuales se adaptan a la transformación aplicada, por lo que no importa que tan rebuscada es la transformación, siempre se generará una imagen correcta de salida.
- **Conclusiones:** Se puede decir que son transformaciones lineales, porque nunca una transformación me llevó a poner 2 veces un valor en el mismo pixel (elemento de la matriz). Además esto se puede ver claramente al notar que las matrices resultantes no aparecen con filas desaparecidas.

### 4. Gráfica de vectores

- **Software:** En el desarrollo del software para la gráfica de vectores, se utilizaron las librerías *numpy*, *pylab* y otras funciones estándar de Python. *Numpy* se ocupó para aprovechar sus capacidades de manejo de matrices y arrays, y *pylab* para poder mostrar un gráfico. Las funciones ejecutan algoritmos lo más atómicos posibles, para así evitar tiempo innecesario de cómputo. Para la obtención de vectores aleatorios, se ocupan números entre -10 y 10, para así evitar números excesivamente grandes o números excesivamente pequeños, ya que la función *random()* solo genera valores entre 0 y 1, y la función *numpy.empty()*, entrega valores excesivamente pequeños.
- **Conclusiones:** Luego de ir probando con distintos valores, se puede notar, que entre más grande es el número de vectores, se va generando la forma de un círculo, con un radio *norm*. Esto es congruente con la forma de las p-normas con  $p=2$ , la cual es un círculo de radio 1.  
Acá con un ejemplo con  $n=1000$  y  $norm = 2$ :

