

# Reporte semanal: Agosto 25-Septiembre 14 ~~Agosto 31~~

Jorge Ballote

23 de septiembre de 2021

## 1. Introducción

A continuación se presenta el reporte de resultados de la tercera semana.

## 2. Objetivos de la semana

Esta semana fue destinada a lo siguiente.

1. Algoritmo genético para la optimización de hiper parámetros.
2. Transformaciones a las series de tiempo para la investigación de diferentes patrones

Los objetivos fueron cumplidos. En las siguiente secciones, se analizan los resultados obtenidos.

## 3. Modificaciones al algoritmo

Se implementó una modificación al algoritmo inicial para transformar las traves con funciones trigonométricas. Además, se implementó un algoritmo genético para optimizar hiperparámetros.

### 3.1. Transformaciones de series de tiempo

En el artículo se describe que se han obtenido buenos resultados, transformando las series de tiempo con funciones trigonométricas. La lógica detrás de esto, consiste en que al aplicar una transformación, las traves lucirán muy diferentes a las originales, y en teoría nuestra grid debería ser capaz de detectar patrones nuevos con estas nuevas traves.

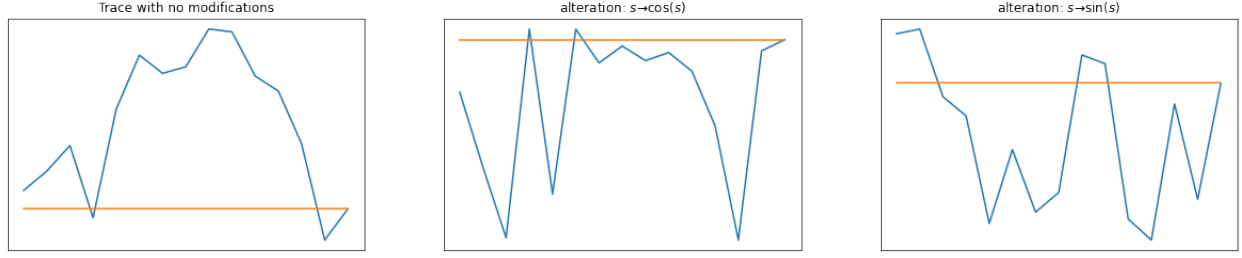


Figura 1: Diferencias entre traces con diferentes operadores aplicados.

Una pregunta importante es, ¿Se aplica la función trigonométrica a cada trace o a los precios en el tiempo?

Supongamos que tenemos una serie de tiempo  $a_1, \dots, a_n$  y un valor siguiente  $a_{n+1}$ . Las traces se obtienen normalizando con el punto final. Es decir

$$b_i = \frac{a_i}{a_n} - 1. \quad (1)$$

De modo que siempre ocurre que  $b_n = \frac{a_n}{a_n} - 1 = 0$ . Esta es una propiedad que nos gustaría preservar. Por otro lado, para saber si estamos en presencia de una trace ganadora o perdedora, lo que se hace es sencillamente comparar si  $a_{n+1} > a_n$  o biseversa. Llamaremos label entonces al siguiente valor:

$$label = \frac{a_{n+1}}{a_n} - 1 \quad (2)$$

También es importante que nuestra label se preserve usando los valores sin modificar (Es decir, antes de aplicar algún operador a la serie de tiempo). De modo que dado un operador  $f : \mathbb{R} \rightarrow \mathbb{R}$ , nuestra trace modificada esta dada por

$$c_i = \frac{f(a_i)}{f(a_n)} - 1, \quad label = \frac{a_{n+1}}{a_n} - 1. \quad (3)$$

Las funciones implementadas fueron, sin, cos y tan. Los resultados se presentaran en la siguiente sección, después de haber sido optimizados con nuestro algoritmo genético

## 3.2. Optimización de hiperparámetros con un algoritmo genético

### 3.2.1. Introducción a algoritmos genéticos

Describiremos brevemente lo que es un algoritmo genético, para más información, consultar a [2].

El algoritmo implementado, está inspirado en el proceso natural de la evolución. Nuestro conjunto de hiperparámetros por optimizar se conocen como cromosomas en este contexto. Y cada hiperparámetro es un gen.

Se tiene una población de soluciones inicial conocida como generación 0. A partir de ahí, usamos los elementos de cada generación para crear a la siguiente. Los genes tienen más posibilidades de

preservarse conforme mejor resultados proporcionen, es decir, mientras más valor tenga una Fitness function. Usualmente se usa la accuracy, pero en nuestro caso usaremos  $\text{criterio2} = \min(5p, 1)(2a - 1)$ .

Para dar lugar a la evolución es necesario el operador *crossover* que combina genes de dos miembros de la población, para obtener un nuevo cromosoma. Además de eso, se procura la diversidad de soluciones, utilizando un operador de mutación, que puede seleccionar una solución y modificarla ligeramente (Cambiarle un parámetro en nuestro caso).

En algunas ocasiones, se implementa el concepto de elitismo, que consiste en forzar a las mejores soluciones a permanecer siempre en la población.

## 4. Metodología

Corrimos nuestro algoritmo genético durante 20 generaciones por experimento. Con una población de tamaño 30. La probabilidad de crossover fue seleccionada de 0.9 y la de mutación de 0.5. Se utilizó un algoritmo elitista, donde las 5 mejores soluciones permanecían siempre como miembros de la población. Tal como se describió, la métrica que nos interesa es  $\min(5p, 1)(2a - 1)$ , de modo que esa será la métrica a optimizar en nuestro algoritmo genético. En particular, tomando  $p$ : 'part above' y  $a$ : 'acc above'

Para la presentación de resultados, usaremos 2 formas de evaluar nuestro algoritmo.

1. **Desempeño en los últimos días:** Simplemente, ¿Si hubiésemos usado nuestro algoritmo en los últimos  $d$  días, ¿Cuántas veces hubiésemos acertado?
2. **K-Cross Validation** El k-cross validation consiste en particionar nuestro conjunto de datos en  $K$  partes diferentes. De modo que entrenamos  $k$  veces nuestro modelo, utilizando  $k - 1$  conjuntos de la partición para el entrenamiento y el conjunto restante para la validación. Finalmente, se suele tomar la media de los  $k$  resultados obtenidos

Para la optimización de hiperparámetros se utilizaron  $n - 365$ . Para la prueba del modelo, se utilizaron 365 días de nuestro vector de precios. Cada posible cromosoma de nuestro algoritmo, indujo un conjunto de traces, los cuales fueron utilizados para conseguir la puntuación del  $k$ -cross validation con  $k = 4$ .

Las métricas a considerar serán las utilizadas en el reporte previo. Incluyendo el criterio 2.

## 5. Resultados

Después de 25 generaciones, nuestra Fitness function alcanzó un valor máximo de 0.16403. Al hacer la prueba de los últimos 365 días se obtienen muy buenos resultados, lo cuál sugiere que los parametros obtenidos son capaces de generalizar.

Hiperparámetros		$\Rightarrow$	Resultados	
trace_size	95		<b>criteria2_above</b>	<b>0.232</b>
height	20		accuracy	53.75
width	95		participation	47.39
grid_type	wins		acc_above	61.60
operation_type	W/L		part_above	30.68
alpha_plus	0.825204		acc_below	39.34
alpha_minus	0.686667		part_below	16.71
eliminate_noise_thold	0.000251377			

### 5.1. Alteración: $s \rightarrow \sin(s)$

Hiperparámetros		$\Rightarrow$	Resultados	
trace_size	79		<b>criteria2_above</b>	<b>0.219</b>
height	4		accuracy	50.70
width	79		participation	96.71
grid_type	wins		acc_above	60.97
operation_type	W/L		part_above	44.93
alpha_plus	0.00423783		acc_below	41.79
alpha_minus	0.131058		part_below	51.78
eliminate_noise_thold	0.0091428			

En el proceso de entrenamiento, obtuvo en promedio **criteria2** = 0.123. En la prueba de los últimos 365 días, se obtuvo que **criteria2** = 0.219.

### 5.2. Alteración: $s \rightarrow \cos(s)$

Hiperparámetros		$\Rightarrow$	Resultados	
trace_size	97		<b>criteria2_above</b>	<b>0.133</b>
height	52		accuracy	50.99
width	97		participation	96.71
grid_type	earnings		acc_above	56.69
operation_type	W/T		part_above	69.58
alpha_plus	0.171751		acc_below	36.36
alpha_minus	0.0240919		part_below	27.12
eliminate_noise_thold	0.0394345			

En el proceso de entrenamiento, obtuvo en promedio `criteria2_above` = 0.123. En la prueba de los últimos 365 días, se obtuvo que `criteria2_above` = 0.133.

### 5.3. Alteración: $s \rightarrow \tan(s)$

Hiperparámetros		⇒	Resultados	
trace_size	45		<b>criteria2_above</b>	<b>0.133</b>
height	95		accuracy	50.86
width	45		participation	95.34
grid_type	wins		acc_above	56.66
operation_type	W/L		part_above	65.75
alpha_plus	0.0358503		acc_below	37.96
alpha_minus	0.103658		part_below	29.58
eliminate_noise_thold	0.0391943			

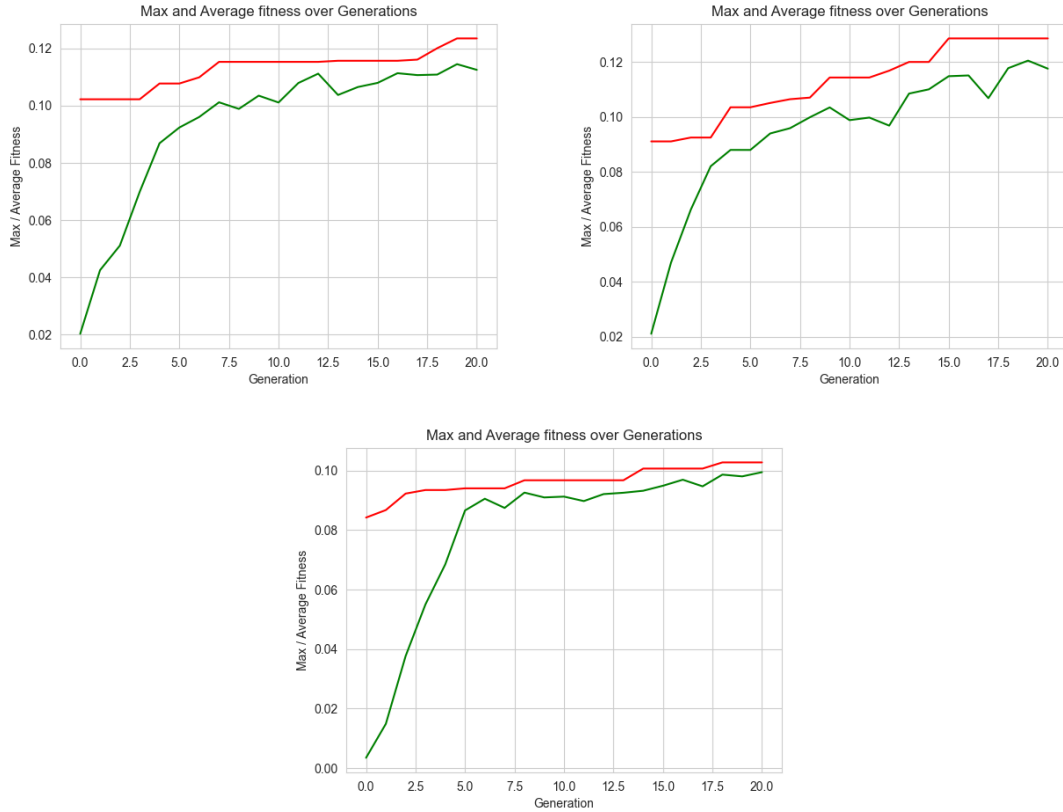


Figura 2: Gráficas de evolución con los operadores sin, cos, tan, respectivamente. La línea roja representa el valor máximo de la fitness function y la línea verde es la media de la población.

## 6. Conclusiones

Pese a que no se documentaron los experimentos anteriores, es importante destacar que sin la implementación del  $k$ -cross validation, la optimización resulta inútil. Lo importante no es que obtenga resultados muy buenos en nuestro conjunto de entrenamiento, sino que pueda generalizar con datos que no ha visto.

Por otro lado, en términos generales, la implementación de las funciones trigonométricas no parecen aportar demasiado aunque cabe destacar que la función  $\sin(x)$  parece tener resultados similares. Tal vez puedan resultar útiles en caso de querer tener varias señales.

Los algoritmos genéticos demostraron ser muy eficientes para optimizar hiperparámetros. En la mayoría de los casos, los resultados en el conjunto de prueba incluso superaban a los del conjunto de entrenamiento. Esto sólo sugiere que se generaliza bien, pero no debe asumirse que el desempeño será mejor siempre en el conjunto de prueba.

Algo interesante es que en la práctica si nuestra fitness function se basa en `criteria2_above`, los resultados mejoran con respecto a una fitness function basada en el simple `criteria2`, cuando en teoría ésta última fitness function, debería optimizar simultáneamente el `criteria2_above` y el `criteria2_below`.