Aprendizaje Evolutivo: Maximización de razón de Sharpe Ratio para un portafolio de inversión

Karla Orozco
Jonathan Zapata
Juan Esteban Chavarria
Juan Fernando Gallego

Maestría en Ciencias de los Datos y Analítica,
Aprendizaje Automático Avanzado, 2022-1,
Universidad Eafit, Medellín, Colombia.

Resumen

Un portafolio o cartera de inversión es el conjunto de activos financieros en el que un inversionista tiene distribuido su capital invertido. Un enfoque muy conocido para realizar esta distribución es maximizando la razón de Sharpe Ratio, basado en la teoría de Markowitz, desarrollada en 1952 [1]; que indica el retorno esperado en exceso de la tasa libre de riesgo por unidad de riesgo total. Por tanto la distribución de dinero en las acciones a invertir que maximiza esta razón, se le conoce como portafolio óptimo; el cual, ofrece el mejor retorno esperado para un nivel dado de riesgo. Para realizar esta optimización se propone utilizar un algoritmo genético, propuesto por John Holland a mediados de los 70; dado que este es un algoritmo de búsqueda que emula la evolución biológica en el computador, siguiendo un proceso evolutivo inteligente sobre individuos, donde su objetivo final es encontrar la mejor solución posible, a partir de soluciones escogidas con un componente aleatorio pero sesgado a favor de las mejores soluciones.

1. Introducción

La optimización de portafolios es un tema de interés para los inversionistas, donde su objetivo es encontrar una cartera de inversión óptima con respecto a la relación retorno-riesgo. Un enfoque muy conocido para realizar esta distribución es maximizando la razón de Sharpe Ratio, basado en la teoría de Markowitz desarrollada en 1952 [1]; que indica el retorno esperado en exceso de la tasa libre de riesgo por unidad de riesgo total. Y es el enfoque que se pretende optimizar en este articulo:

La formula de la razón de Sharpe ratio (S) para un portafolio de inversión (ptf) esta dada por:

$$S = \frac{R_{ptf} - R_f}{\sigma_{ptf}}$$

Donde:

R_{ptf} es el retorno esperado del portafolio de inversión, dado por el producto punto entre los pesos a invertir en n activos financieros (W) y el retorno esperado de cada uno de estos activos (R):

$$R_{ptf} = (W_i, \ldots, W_n) \cdot (R_i, \ldots, R_n)$$

R_f es la tasa libre de riesgo.

Y σ_{ptf} es el riesgo esperado del portafolio de inversión, calculado como la multiplicación matricial entre 3 componentes: 1. los pesos a invertir (W), 2. la matriz de covarianza de los n activos financieros (af); que indica el grado de variación conjunta entre cada par de activos, y 3. los pesos a invertir transpuestos (W^T), así:

$$\sigma_{ptf} = (W_i, \ldots, W_n) \times \Sigma_{af} \times (W_i, \ldots, W_n)^T$$

Para realizar esta optimización se propone utilizar un algoritmo genético, propuesto por John Holland a mediados de los 70 [2]; dado que este es un algoritmo de búsqueda que emula la evolución biológica en el computador, siguiendo un proceso evolutivo inteligente sobre individuos, donde su objetivo final es encontrar la mejor solución posible, a partir de soluciones escogidas con un componente aleatorio pero sesgado a favor de las mejores soluciones.

Procedimiento general de un algoritmo genético

Ilustración basada del articulo [3]:

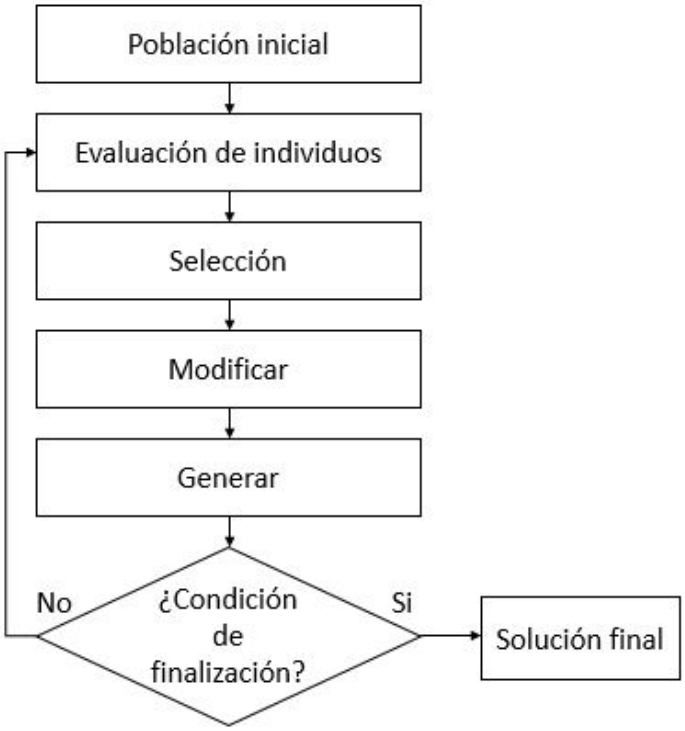

Figura 1: Macro-algoritmo del algoritmo genético

El proceso ilustrado en la figura 1 se detalla así: Inicialmente se genera una población inicial, compuesta por posibles soluciones al problema en cuestión; conocidos como *cromosomas*, luego estos se evalúan con respecto a una función que se busca optimizar; conocida como *función fitness*. Lo siguiente para conseguir la próxima generación; en la forma estándar del algoritmo es, realizar la selección por el azar de la rueda de la ruleta, donde los cromosomas son asignados en un espacio proporcional a su aptitud resultante de la evaluación, por lo tanto será más probable que sean seleccionados los más aptos. Lo que sigue es producir nuevos cromosomas modificando la población seleccionada; conocida como *padrotes*, por medio de operadores genéticos (se explicarán abajo) para proseguir con la generación de una población compuesta por la existente y los individuos nuevos generados. Por último se evalúa si se cumple el criterio de convergencia o de finalización; donde si no, se regresa desde el proceso de evaluación o en caso contrario se obtiene el resultado de la mejor solución. Cabe resaltar que en el proceso de obtener la próxima generación se puede considerar un criterio *elitista*; el cual permite que un porcentaje de los individuos más aptos de la población inicial trascienda a la siguiente generación, sin cambios, garantizando que la calidad de la solución no disminuya de una generación a la otra siguiente. Aunque esto puede incrementar la tasa de convergencia, tener un porcentaje alto de élites puede hacer que el algoritmos se que atrapado en un óptimo local, por tanto este valor si se va a incluir, suele ser bien pequeño. Y no siempre es necesario, a veces incluso deteriorar el resultado.

1.1. Operadores genéticos clásicos

Estos suelen estar presentador como una representación binaria de los cromosomas; sus componentes son conocidos como *genes*.

1.1.1. Cruce

Permite representar el proceso de apareamiento natural, intercambiando pedazos de información aleatoriamente entre diversos individuos para generar unos nuevos. A continuación su ilustración:

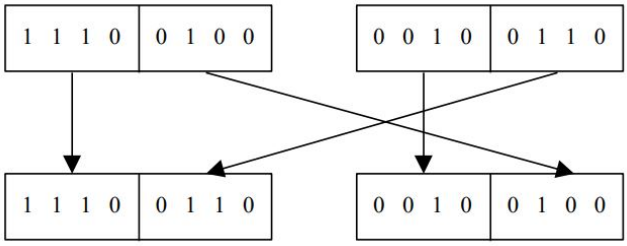

Figura 2: Operador de cruce

1.1.2. Mutación

Permite simular el proceso evolutivo que ocurre en los individuos, cuando cambian su estructura genética; son seleccionados aleatoriamente genes del cromosoma para modificarlos, también de manera aleatoria; produciendo cambios en la estructura del individuo. A continuación su ilustración:

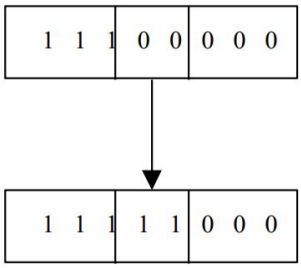

Figura 3: Operador de mutación

Lo que sigue del articulo, presenta la metodología empleada para realizar el proceso mencionado y los resultados obtenidos.

2. Metodología

Para el desarrollo de este trabajo se emplea el lenguaje de programación *Python* en el ambiente de *JupyterLab* y se sigue la metodología CRISP-DM [4]; clásica para abordar un problema de analítica, la cual se compone de 6 etapas que dan un orden lógico al proceso; desde el entendimiento del negocio y los datos, seguido por su preparación, luego modelado y evaluación, donde si no se alcanzan los objetivos del negocio, se regresa al inicio y en el otro caso, llega hasta su despliegue en producción. El alcance de este trabajo no incluye la última etapa. Aquí su ilustración:

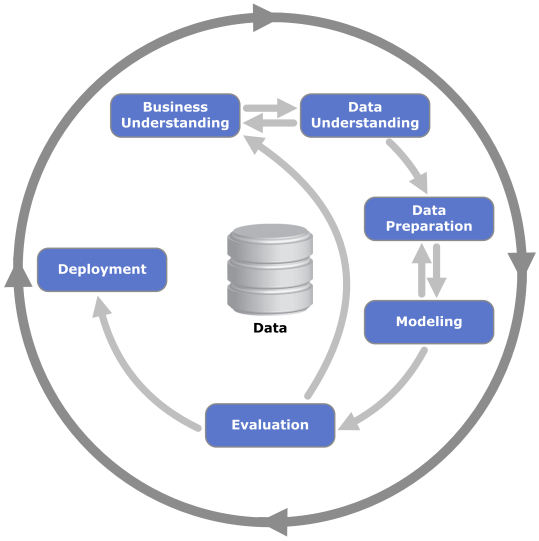

Figura 4: Metodología CRISP-DM

2.1. Entendimiento del negocio

Ya habiendo definido el problema que se busca resolver, se debe definir que activos financieros conformarán el portafolio a optimizar. Para esto los autores consideran seleccionar las acciones de las primeras 10 empresas por capitalización de mercado (valor total de todas las acciones de una empresa) del índice S&P 500; el cual es un índice bursátil reconocido que sigue el desempeño de las 500 compañías más grandes que cotizan en la bolsa de valores de los Estados Unidos.

Se descargan los precios de cierre ajustado mensuales de los últimos 5 años (2017-01-01 hasta 2021-12-31) de las acciones de las empresas elegidas. Para esto se utiliza el paquete *yfinance* de Python, indicando los *tickers* de estas acciones; código de identificación de un activo financiero.

Top 10 de empresas del S&P 500 por capitalización de mercado:

No.	Compañia	Ticker
1.	Tesla Inc	TSLA
2.	Amazon.com Inc	AMZN
3.	Alphabet Inc Class A	GOOGL
4.	Alphabet Inc Class C	GOOG
5.	Microsoft Corporation	MSFT
6.	Apple Inc	AAPL
7.	Berkshire Hathaway B	BRK-B
8.	IPG Photonics Corporation	IPGP
9.	PVH Corp	PVH
10.	Penn National Gaming Inc	PENN

2.2. Entendimiento de los datos

La estructura inicial de los datos de los precios históricos de las acciones mensuales descargados en el rango indicado, se presenta a seguidamente:

Date	AAPL	AMZN	BRK-B	GOOG	GOOGL	IPGP	MSFT	PENN	PVH	TSLA
2017-01-01	28.52	823.48	164.14	796.79	820.19	114.99	59.99	13.78	93.25	50.39
2017-02-01	32.20	845.04	171.42	823.21	844.93	118.30	59.36	14.47	91.05	50.00
2017-03-01	33.91	886.54	166.68	829.56	847.80	120.70	61.48	18.43	102.89	55.66
2017-04-01	33.91	924.99	165.21	905.96	924.52	126.32	63.91	18.48	100.46	62.81
2017-05-01	36.06	994.62	165.28	964.86	987.09	139.04	65.19	19.33	105.36	68.20

Figura 5: Precios mensuales de las acciones

Se evalúa que los datos nos tengan valores nulos:

```
AAPL     0
AMZN     0
BRK-B    0
GOOG     0
GOOGL    0
IPGP     0
MSFT     0
PENN     0
PVH      0
TSLA     0
dtype: int64
```

Figura 6: Evaluar nulos

Tal como se aprecia, se cuenta con todos los datos las 10 acciones en el rango a analizar.

2.3. Preparación de los datos

Con el fin de computar la razón de Sharpe Ratio a optimizar, se organizan los datos, como se detalla a continuación:

Primero se calculan los retornos históricos mensuales de los precios, para luego computar el retorno esperado por acción, comprendido como el promedio histórico de sus retornos:

Date	AAPL	AMZN	BRK-B	GOOG	GOOGL	IPGP	MSFT	PENN	PVH	TSLA
2017-02-01	0.12	0.03	0.04	0.03	0.03	0.03	-0.01	0.05	-0.02	-0.01
2017-03-01	0.05	0.05	-0.03	0.01	0.00	0.02	0.04	0.24	0.12	0.11
2017-04-01	-0.00	0.04	-0.01	0.09	0.09	0.05	0.04	0.00	-0.02	0.12
2017-05-01	0.06	0.07	0.00	0.06	0.07	0.10	0.02	0.04	0.05	0.08
2017-06-01	-0.05	-0.03	0.02	-0.06	-0.06	0.04	-0.01	0.10	0.08	0.06

Figura 7: Retornos mensuales de las acciones

	mean_returns
AAPL	0.030974
AMZN	0.023703
BRK-B	0.010165
GOOG	0.021859
GOOGL	0.021388
IPGP	0.006838
MSFT	0.029184
PENN	0.022460
PVH	0.002268
TSLA	0.051581

Figura 8: Retorno esperado mensual por acción

Luego se calcula la matriz de covarianza de los retornos de las acciones (Σ_{af}):

	AAPL	AMZN	BRK-B	GOOG	GOOGL	IPGP	MSFT	PENN	PVH	TSLA
AAPL	0.0071	0.0036	0.0018	0.0024	0.0024	0.0037	0.0026	0.0057	0.0051	0.0074
AMZN	0.0036	0.0061	0.0015	0.0027	0.0026	0.0036	0.0028	0.0038	0.0039	0.0045
BRK-B	0.0018	0.0015	0.0025	0.0017	0.0017	0.0022	0.0013	0.0044	0.0047	0.0028
GOOG	0.0024	0.0027	0.0017	0.0041	0.0041	0.0020	0.0021	0.0052	0.0047	0.0042
GOOGL	0.0024	0.0026	0.0017	0.0041	0.0040	0.0022	0.0021	0.0053	0.0047	0.0041
IPGP	0.0037	0.0036	0.0022	0.0020	0.0022	0.0139	0.0021	0.0085	0.0089	0.0033
MSFT	0.0026	0.0028	0.0013	0.0021	0.0021	0.0021	0.0026	0.0033	0.0032	0.0040
PENN	0.0057	0.0038	0.0044	0.0052	0.0053	0.0085	0.0033	0.0426	0.0181	0.0145
PVH	0.0051	0.0039	0.0047	0.0047	0.0047	0.0089	0.0032	0.0181	0.0212	0.0086
TSLA	0.0074	0.0045	0.0028	0.0042	0.0041	0.0033	0.0040	0.0145	0.0086	0.0300

Figura 9: Matriz de covarianza de los retornos de las acciones

Posteriormente, se descarga la tasa libre de riesgo a la fecha actual del análisis (2021-12-31), comprendida como el rendimiento anual de los bonos del Tesoro US a 10 Años, utilizando también el paquete *yfinance* de Python. La tasa obtenida a esa fecha es: 1,512 Efectiva Anual. Para emplearla, se lleva a términos periódicos (mensuales en este caso).

Así las cosas, ya se cuenta con todos los componentes para computar la Razon de Sharpe Ratio, optimizando las proporciones o pesos de invertir en las acciones; tal que, maximicen este indicador.

2.4. Modelado

Para esta etapa se utiliza la función *geneticalgorithm* de Python, para implementar el algoritmo genético, pero pasa esto se debe construir la función *fitness* a evaluar que se busca optimizar:

El sumatorio de los pesos (W) debe ser igual a 1. Para cumplir con esta restricción, cada peso se divide sobre el sumatorio de sus valores, para obtenerlos como proporción. El rango de valores que pueden tomar estos pesos a optimizar se define entre $[-1, 1]$, se pueden tener posiciones de venta o de compra. Y adicional se incluye una restricción para evitar la indeterminación: $\sum_{i=1}^{n} W_i = 0$. Esto se considera que no es necesario, dado que el dominio para evaluar las variables de decisión es de números reales y por tanto el caso de que tomen el valor especifico de cero no es muy factible. No obstante, nada se pierde con esta precaución. Y si fuera el caso de que la suma de todos los pesos es cero, se asigna una distribución de pesos iguales, considerada la más sencilla para diversificar los recursos a invertir; también conocida como distribución naïve.

Por tanto la función fitness es:

$$\begin{aligned} \max_{W} \quad & S = \frac{R_{ptf} - R_f}{\sigma_{ptf}} \\ \text{donde} \quad & R_{ptf} = W \cdot R \\ & \sigma_{ptf} = W \times \Sigma_{af} \times W^T \\ & \text{s.a.} \sum_{i=1}^{n} \frac{W_i}{\sum_{i=1}^{n} W} = 1 \end{aligned}$$

Parámetros del algoritmo genético

Según lo índica la documentación de la función a utilizar, se tiene:

- max_num_iteration: Criterio de parada, Si es Ninguno, el algoritmo establecer el número máximo de iteraciones en función de la dimensión, limites y tamaño de la población. El valor por defecto es Ninguno. Y según esta documentación es el recomendado.
- population_size: Número de posibles soluciones en cada iteración. El valor por defecto es 100.
- mutation_probability: Posibilidad de que cada gen en cada solución individual sea reemplazado por un valor aleatorio. El valor por defecto es 0.1.
- elit_ratio: número de élites en la población. El valor por defecto es 0.01. Si se establece en cero, se implementa un algoritmo genético estándar en lugar de un elitista.
- crossover_probability: Posibilidad de que una solución existente pase su genoma a nuevas soluciones de prueba. El valor por defecto es 0.5
- parents_portion: Parte de la población ocupada por los miembros de la generación anterior. El valor por defecto es 0.3.
- crossover_type: Tipo de cruce. El valor por defecto es cruce uniforme.
- max_iteration_without_improv: Si el algoritmo no mejora la función objetivo sobre el número de iteracionres sucesivas determinadas por este parámetro, entonces se detiene e informa la mejor solución encontrada antes de que se cumple el max_num_iterations. El valor por defecto es Ninguno.

Con el objetivo de afinar los parámetros del algoritmo para su despliegue, se procede a aplicar *RandomSearch*.

Para esto se construye una función que evalúa el algoritmo, realizando n iteraciones (se definen 100 iteraciones), combinando aleatoriamente los parámetros a variar, según bien sea una lista o una distribución de sus posibles valores. Para la distribución se toma la uniforme, dado que en principio no se cuenta con una indicación de sus correctos valores.

Para este trabajo, los parámetros a sensibilizar, posibles valores y sus tipos son:

Parámetros	**Posibles valores**	**Tipo**
population_size	(De 50 a 300)	Distribución uniforme
mutation_probability	(De 0.1 a 1)	Distribución uniforme
elit_ratio	[0,0.01,0.02]	Lista
crossover_type	(De 0.1 a 1)	Distribución uniforme
parents_portion	(De 0.1 a 1)	Distribución uniforme

2.5. Evaluación

El mejor resultado obtenido por el algoritmo genético con la evaluación de RandomSearch es:

Parámetros del algoritmo genético:

```
{'max_num_iteration': None,
 'population_size': 196.97765612984807,
 'mutation_probability': 0.47777054589344514,
 'elit_ratio': 0,
 'crossover_probability': 0.9441574740149382,
 'parents_portion': 0.5290027266308917,
 'crossover_type': 'uniform',
 'max_iteration_without_improv': None}
```

Valor optimo de sharpe ratio:

```
0.6381028640876919
```

Variables de decisión como proporción:

```
array([ 0.14157838, -0.25434807,  0.04538768,  0.94733566, -0.87002871,
        0.00403196,  1.15268247,  0.03636399, -0.22292326,  0.01991992])
```

```
[24]: Text(0, 0.5, 'Objective function')
```

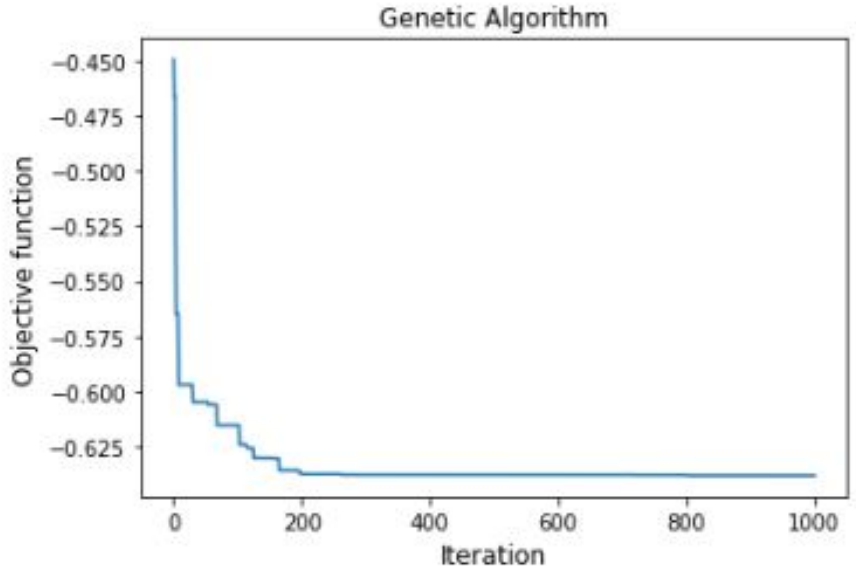

Figura 10: Resultado del algoritmo genético con evaluación de RandomSearch

3. Conclusiones

Se concluye que los algoritmos genéticos son una alternativa interesante y poderosa para trabajos de optimización, donde a partir de su proceso de iteración consigue converger a una solución óptima.

Además, se aprecia que es de gran importancia el proceso de afinar los parámetros del algoritmo para abordar el problema de interés, donde la mejor solución obtenida, fue con parámetros considerablemente distintos a los que tiene el algoritmo genético por defecto.

Referencias

[1] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7:77–91, 1952.

[2] John H. Holland. Genetic algorithms computer programs that "evolve" in ways that resemble natural selection can solve complec problems even their creators do not fully understand. 1992.

[3] Yu Lean, Shouyang Wang, and Kin Keung Lai. Portfolio optimization using evolutionary algorithms. *Reflexing Interfaces: The Complex Coevolution of Information Technology Ecosystems*, pages 235–245, 2008.

[4] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. 2000.