INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Proyecto Final

Algoritmos Numéricos por Computadora

Profesor Ángel Fernando Kuri

11 de diciembre del 2020

Francesca Perrone Jiménez 179682

Brandon Francisco Hernández Troncoso 154328

Emilia Sofia Spinola Campos 172103

Miguel Ángel Cifuentes Jiménez 168274

Fundamentos teóricos

El propósito del Exchange Algorithm, el algoritmo de aproximación, es expresar el comportamiento de una variable como una función de una serie de variables independientes. Esto se puede ver de la siguiente forma.

$$y = f(v_1, v_2, \ ..., \ v_n) \ = f(\widehat{v}) \ \ *$$

*Llamaremos a esta nuestra función 1

En esta ecuación y es una variable dependiente y v funciona como un set de variables independientes que expresan a la variable y en función de f.

La forma en la que se define a la aproximación por realizar es la siguiente:

$$y = c_1X_1 + c_2X_2 + ... + \ c_mX_m$$

Aquí c representa cada uno de los diferentes coeficientes, m representa el número de coeficientes que se desean y X_i denota las combinaciones de variables independientes $(X_i = f_i(\widehat{v}))$. Dependiendo de la forma en como estas combinaciones se definan, se pueden obtener diferentes aproximaciones.

El método asume que existe una muestra de tamaño S tal que para cada set de variables independientes $(\widehat{v})$ existe un valor conocido de la variable dependiente $(f(\widehat{v}))$.

Tabla de Ejemplo (obtenida de la presentación de clase):

v1	v2	v3	v4	v5	f
0.2786	0.2319	0.5074	0.9714	0.5584	0.4938
0.2429	0.4855	1.0000	0.8429	0.4416	0.8580
0.4929	0.9710	0.8676	1.0000	0.7411	0.9136
0.8429	1.0000	0.4485	0.9143	1.0000	0.8704
0.4357	0.9130	0.8309	0.6500	0.8579	0.7037

Todos los elementos de la muestra se llaman objetos, y existen N objetos en la muestra. También asumimos que la muestra ha sido seleccionada de forma apropiada, de tal manera que haya suficiente información contenida en la muestra para que la aproximación pueda ser una representación útil del sistema que estamos estudiando. En nuestro caso se estudia el sistema de contagios por COVID-19.

El objetivo del algoritmo de aproximación (Exchange Algorithm) es encontrar los valores de los coeficientes en nuestra función 1, de tal forma que los valores aproximados logren minimizar la diferencia entre los valores conocidos de la variable dependiente f y los valores calculados de la función 1 de todos los objetos en la muestra. Para determinar los coeficientes de aproximación necesitamos definir un error.

Para encontrar el menor error utilizamos la aproximación minimax. Definimos el algoritmo de aproximación de la siguiente forma.

$$\varepsilon = max\,(\,\varepsilon_1,\ \varepsilon_2,\ ...,\ \varepsilon_m\,) \text{ donde } \varepsilon_1 = |f_i - y_i|$$

Aquí f_i representa el valor conocido del objeto y y_i representa el valor del objeto que el algoritmo obtiene cuando los X_i se introducen en nuestra función 1. Es decir, encontramos el polinomio de aproximación y lo comparamos con el valor real f_i, la diferencia entre estos es el error.

El ajuste mínimo, el ajuste que tiene el error más pequeño, es aquel en el cual los errores de aproximación son todos del mismo tamaño absoluto (todos los errores dan un valor absoluto igual entre sí), y esto se consigue por medio de la aproximación minimax (el mínimo de los errores máximos). Recordemos que esto se obtiene como resultado de encontrar el error utilizando la Regla de Cramer. Para obtener el signo del error utilizamos el teorema que se explicó en clase, que establece que si los cofactores de una columna de un determinante se multiplican por los elementos de una columna distinta y se suman el resultado debe ser cero.

Los N objetos con los que contamos son separados en dos sets, un set interno y un set externo. El set interno consiste de M objetos y el set externo consiste de $N - M$ objetos. La condición minimax se obtiene por todos los objetos del algoritmo y los coeficientes obtenidos serán aquellos que representan la mejor aproximación a nuestra función 1.

Ahora hablaremos del algoritmo de ascenso, el cual se utiliza para encontrar los dos errores que nos interesan. El primer error que nos interesa se obtiene del valor absoluto del error aproximado más grande del set interno. Este error se denomina ε_θ. El segundo error que nos interesa se obtiene del valor absoluto del error aproximado más grande del set externo, a este se le denomina ε_Φ.

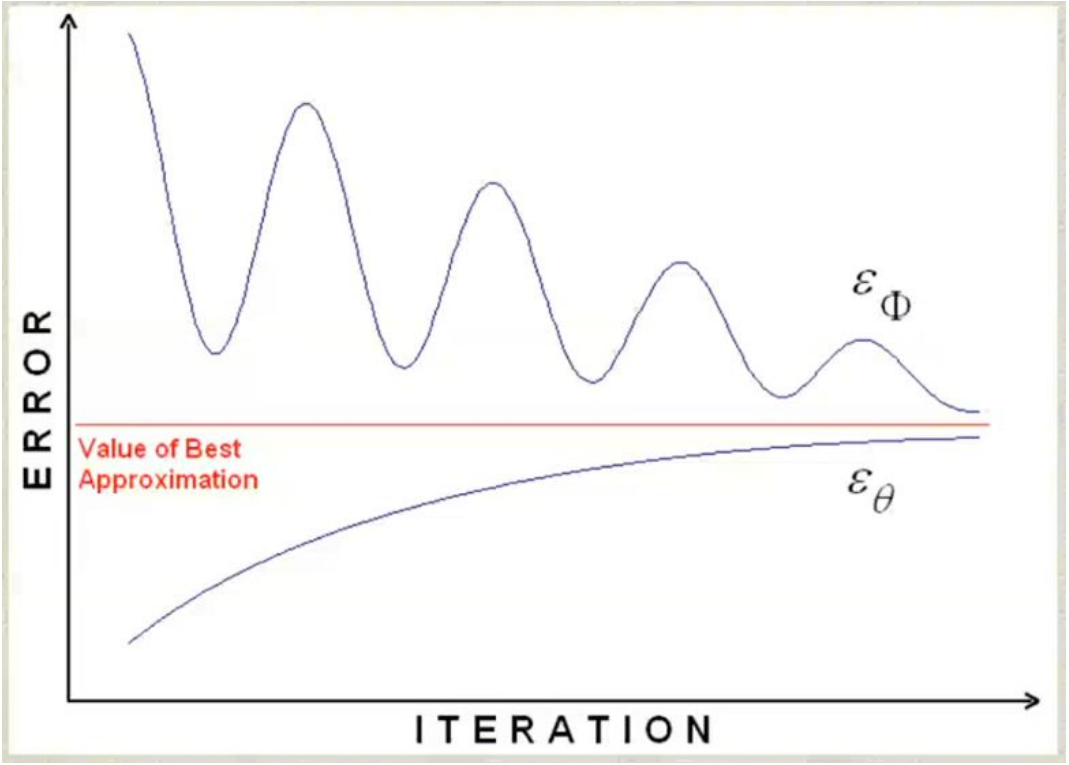
En primer lugar se calcula el error interno y luego se calcula el error externo. Podemos llegar a observar que para obtener el error externo inicialmente evalúan valores grandes y conforme avanzamos este error va decreciendo de forma

asintótica pero no monótona. Por otro lado el error interno comienza con un valor pequeño e incrementa conforme avanzamos de forma monótona.

Ambos errores se aproximan entre sí hasta alcanzar la condición de convergencia teórica:

$$\varepsilon_{\theta}(t) \geq \varepsilon_{\Phi}(t)$$

Cuando se cumple la condición de convergencia estamos encontrando el error mínimo del set entero porque el error interno será igual de bueno que los errores externos.



Gráfica 1: representación de condición de convergencia entre ε_{θ} y ε_{Φ}

Si logramos encontrar la mejor aproximación para el set interno, por definición estamos encontrando el error más pequeño en ese set, y la aproximación polinomial correspondiente de la iteración t (es decir $y_t(X)$) es la cual tiene los coeficientes que minimizan la diferencia entre el set de datos observados y la nuestra función 1 en t.

Si los valores que obtenemos de $y_t(X)$ para los objetos en el set externo nos dan un valor de error igual o menor que el error del set interno (condición de convergencia) , entonces podemos decir que $y_t(X)$ es la mejor solución general.

Si, los errores que obtenemos de los valores de $y_t(X)$ en el set externo son mayores que los valores que obtenemos de los $y_t(X)$ del set interno, entonces

tenemos que intercambiar un objeto interno con un objeto externo y repetir todo el proceso.

Este nuevo set interno nos va a dar un error nuevo $\varepsilon_{\theta}(t+1)$ y este nuevo error necesariamente va a ser igual o más grande que el error pasado (por ser monótono). Asimismo, coeficientes de $y_{t+1}(X)$ nos van a dar un nuevo error $\varepsilon_{\Phi}(t+1)$ que puede o puede no ser más chico que el error pasado.

Como el error interno siempre está creciendo, eventualmente vamos a llegar a un punto en el que los coeficientes de y serán los mejores para el set interno y el set externo.

Programación

Para generar una solución al problema planteado y obtener los coeficientes que se adaptaran más a los datos de prueba y entrenamiento, se decidió hacer una automatización del código proporcionado por el profesor. La idea detrás de la optimización del código es poder realizar pruebas para diferentes grados máximos de cada variable, buscando de esta manera obtener la mejor combinación de potencias y coeficientes que minimicen el error para los resultados de los datos de prueba.

El primer paso de la implementación fue generar los conjuntos de datos de entrenamiento y prueba para obtener los resultados del proyecto. El archivo del cual se obtuvieron dichos conjuntos fue el archivo *SSAMP300.xls*, el cual es una muestra de 300 datos de una base de datos con 134,814 registros originales sobre información de personas y sus condiciones de salud ante un diagnóstico (positivo, negativo o no especificado) de COVID19. De la muestra utilizada, se dividieron los datos en dos archivos diferentes: un archivo de datos para entrenamiento que fue llamado *customTrain.xls*, y un archivo para la realización de las pruebas llamado *customTest.xls*.

El segundo paso de la implementación fue usar la información generada y almacenada en *customTrain.xls,* el cual contiene los datos con los cuales se buscará generar el modelo que mejor se acomode a los datos de entrenamiento y prueba. El código original inicia con la lectura de los datos de entrenamiento para luego definir el grado máximo que cada variable puede tomar para cada coeficiente del polinomio, sin embargo, como se habló al inicio de esta sección, se hizo una optimización del código, por lo que en vez de especificar un grado máximo diferente para cada variable, se especifica un grado máximo general que pueden alcanzar todas las variables. De esta manera se busca generar la mejor combinación de potencias que se adapten más a los datos de entrenamiento y prueba y que minimicen el error de los conjuntos de datos.

Mediante ciclos se iteró sobre los grados máximos para cada una de las variables independientes, buscando de esta manera realizar permutaciones para los distintos grados en cada variable y generando resultados para cada grado máximo. Realizar este proceso de iteraciones permite cubrir, desde un grado mínimo hasta un grado máximo, todas las posibles combinaciones de grados y generar modelos para cada caso. Los resultados de cada iteración se evalúan en la función *converge*, la cual es una función incluida en el código original proporcionado por el profesor que sirve para verificar si los resultados obtenidos para el modelo generado cumplen ciertas condiciones de convergencia.

Después, con ayuda de los coeficientes obtenidos y las combinaciones de potencias, se realizaron los cálculos de los resultados para los datos de prueba almacenados en el archivo *newCustomTest.xls.* El archivo *newCustomTest.xls* es una copia del archivo *customTest.xls* pero eliminando la columna de resultados, lo cual facilita el procesamiento de los datos de prueba para los modelos generados y, posteriormente, facilita también la comparación entre los resultados obtenidos y los resultados verdaderos. De esta manera, para cada registro de los datos de prueba se realizaron las operaciones matemáticas pertinentes (multiplicaciones, sumas y potencias) para obtener el resultado completo para cada combinación de coeficientes.

Finalmente, se obtuvo el RMSE (Root Mean Square Error) entre los resultados originales del archivo *customTest.xls* y los datos obtenidos y generados con las combinaciones de coeficientes. La combinación de potencias y coeficientes que mejor se adaptara a los datos de entrenamiento y de prueba fueron elegidos a partir de los RMSE calculados, eligiendo como mejor modelo aquel que tuviera un valor menor de RMSE en comparación con todos los modelos generados. Al final los coeficientes resultantes asociados al mejor modelo se guardaron en un archivo llamado *coefResultados.xls* con la intención de estar disponibles para operaciones de interés.

Aplicaciones

Hoy en día, los modelos multivariados tienen diversas aplicaciones dentro de nuestra vida diaria. Podemos encontrar estos modelos dentro de distintos ámbitos del día a día, por ejemplo, viendo algún partido de su deporte favorito, en la construcción de cohetes, puentes, etc.

Por lo general, estos modelos son más reconocidos en los deportes, ya sea de fútbol, fútbol americano o básquetbol. En este caso, hablaremos sobre cómo surgieron estos modelos dentro del béisbol.

Dentro de la Major League of Baseball (MLB), el análisis de los jugadores ha sido fundamental para el desarrollo de distintos modelos realizados por apostadores o

los propios equipos de la Major League. Dentro de todos estos modelos, encontramos que tienen en común distintas variables determinantes para la correcta predicción de los juegos, por ejemplo, cómo han bateado cada bola en la última semana, mes, año o, incluso, durante toda su carrera como jugador profesional. Sobre los strikes que ha tenido y cómo han sido, por ejemplo, si el pitcher es zurdo o diestro o si el bateador lo es; además de cómo se mueven los jugadores dentro del campo.

En el pasado, estos datos eran obtenidos por fans a partir de las tarjetas coleccionables, donde al reverso se encontraban los datos más importantes del jugador (strike, balls, home runs, etc). Sin embargo, en 1980, fue cuando los estadistas se involucraron en el deporte, dándose cuenta de que había un gran negocio gracias a los modelos multivariados y el béisbol pasó a ser un deporte tan analizado.

En la actualidad los estadistas del béisbol han examinado todas las variables que pueden cuantificar y las han asignado a algún valor en particular. Todos estos datos los proporciona la misma MLB, ya que, en la actualidad, el análisis del deporte es fundamental para los equipos. La MLB ofrece información de cada uno de los jugadores de forma gratuita y completa. Por esta razón, los modelos de béisbol se consideran tan justos y transparentes, ya que todos los equipos tienen las mismas posibilidades de realizar modelos o análisis a partir de los mismos datos.

Análisis metodológico

El problema fundamental de este proyecto es encontrar modelos matemáticos de muchas variables (modelos multivariados) que permitan expresar el comportamiento de una variable independiente como una suma de monomios individuales compuestos de variables independientes. Lo primero que se debe hacer es determinar una función que depende de las otras variables independientes, en este caso determinar si una persona es propensa a contagiarse de COVID-19 dadas algunas condiciones de salud.

Para poder generar este modelo, se tiene la condición de contar con suficiente información de tal manera que pueda ser ajustada a un modelo que minimice el error para dichos datos y se ajuste al comportamiento de los mismos. Sin embargo, el modelo no puede ser desarrollado con toda la información disponible puesto que no habría forma de comparar su desempeño y evaluar si efectivamente el error se ha minimizado. Para evitar este problema, se toma la información disponible y se generan dos subconjuntos de datos provenientes de dicha información, a los cuales se les denomina conjunto interno y conjunto externo.

La siguiente acción a realizar es entrenar el algoritmo con toda la información disponible, ya que de esta manera nos aseguramos que todos los datos usados

representan a la población a analizar y permite generar un modelo que se ajuste a dichos datos. Con ayuda del algoritmo de ascenso, se realiza un entrenamiento del algoritmo que nos permite intercambiar datos entre los conjuntos interno y externo con la intención de obtener el mínimo error entre esos datos para cada conjunto. Una vez que se realiza el algoritmo y se cumple la condición de convergencia, se obtiene un polinomio conformado de los grados de las variables independientes y sus respectivos coeficientes.

El algoritmo se divide en dos partes. La primera parte consta de ingresar *n* tuplas, dichos datos nos servirán para expresar el polinomio que estamos buscando. Para este paso, son necesarias *n*+1 tuplas. Una vez que se determinan los *n* coeficientes, obtenemos un elemento que representa el error de aproximación. Este error estará relacionado con un signo, el cual indica si el valor del error es muy grande, es decir, es positivo (el signo sería '+') o muy chico, es decir, negativo (con un signo '-'). La obtención de estos signos se logra mediante los cofactores.

Dado que se busca encontrar una combinación de potencias que minimicen el error para los conjuntos de datos, se debe corroborar que nuestro modelo funciona y se ajusta efectivamente a todos los datos de la población. Para corroborar el funcionamiento y evaluar su desempeño, se toman datos de prueba (test) que representan a la población de donde se obtuvo la muestra pero son datos que el algoritmo no usó para entrenarse. Lamentablemente, aunque algunas combinaciones de coeficientes y potencias minimicen el error en comparación con otras combinaciones, es posible que esa combinación no se ajuste completamente a los datos de prueba por lo que es necesario obtener aquella combinación de potencias que minimice el error pero que se acople de mejor forma a la información de prueba, por lo que se debe realizar una serie de evaluaciones sobre cada modelo generado buscando cumplir estas condiciones.

Resultados

El objetivo del trabajo realizado fue el poder predecir de forma automática si una persona es susceptible o no a contagiarse de COVID-19 dada cierta información sobre condiciones o enfermedades en las personas. Este proceso de predicción se realiza mediante la obtención de un modelo multivariado obteniendo las combinaciones de potencias y valores de coeficientes que minimizan el error entre datos de entrenamiento y prueba para un set de datos de registros relacionados al COVID-19.

La implementación en Python realizada para la solución del problema planteado permitió el poder utilizar cantidades muy grandes de combinaciones que serían imposibles de realizar a mano. El realizar el trabajo de esta forma nos dio la oportunidad de observar el comportamiento de los datos de manera más cercana y así poder determinar los requerimientos para su manipulación.

Gracias a la automatización del proceso de selección de grados y el proceso que se siguió para la obtención de los resultados, se pudo obtener la combinación que otorgaba el menor error para Epsilon Theta y el menor RMSE en comparación con todos los modelos generados. Esta acción permitió reducir el tiempo que se tuviera que invertir en cada iteración cuando se tiene que ingresar los grados máximos que cada variable puede tomar o preguntando e ingresando el nombre del archivo a usar.

Después de un proceso de pruebas se encontró que los valores de grado máximo que se debían considerar para cada variable tendrían que ser 2 y 3, ya que esto nos asegura que el tiempo de procesamiento sea menor y que se cuenta con la cantidad de información necesaria para realizar las actividades en comparación con la información requerida para combinaciones de grados mayores para cada variable.

En el análisis de los resultados que se obtuvieron, se observó que los errores mínimos indicaron que los ajustes a los datos no fueron tan exactos como se esperaba en un inicio, ya que para un grado máximo de 3, usando los conjuntos de datos generados, el RMSE mínimo fue de 0.4. A pesar de ser un valor aceptable, ya que un RMSE mayor a 0.5 indica una pobre habilidad del modelo a ajustar y predecir datos, el propósito de la medida es ser minimizada para determinar un modelo que se ajuste perfectamente a los datos que se le dan para entrenamiento.

Lamentablemente, después de realizar un proceso de pruebas con diferentes conjuntos de datos, nos dimos cuenta que no son suficientes datos para poder obtener un modelo sólido que pueda predecir la información objetivo, por lo que el resultado óptimo depende mucho de la distribución de los datos y de las operaciones del algoritmo de ascenso sobre el intercambio de datos entre los conjuntos internos y externos.

Después de algunas pruebas se obtuvo la siguiente combinación de coeficientes:

c[00000]=-0.24933471010490585	c[10010]=3.3119570792047477
c[00001]=2.2078189365668357	c[10011]=-2.437516885538301
c[00010]=23.00121174644182	c[10100]=10.455042950881095
c[00011]=-23.874846929377956	c[10101]=-7.472189859405564
c[00100]=-0.6976080850953775	c[10110]=-14.267676884522333
c[00101]=-2.0496470441254204	c[10111]=12.118473754312012
c[00110]=-21.80382834716506	c[11000]=0.0021469229748506246
c[00111]=23.716328569920435	c[11001]=2.736223556873169
c[01000]=-0.0011269876832046752	c[11010]=-10.064786551375235
c[01001]=-2.737840023849679	c[11011]=9.265133992334404
c[01010]=-46.50327741536117	c[11100]=-13.04546519726694
c[01011]=47.30224429620281	c[11101]=8.171394450260474
c[01100]=1.4110771749621212	c[11110]=23.11295392978464
c[01101]=2.5509848595574995	c[11111]=-20.177038668410695
c[01110]=45.09256004647704	
c[01111]=-47.11498409357338	
c[10000]=-0.0010291890815343349	
c[10001]=-1.7072603537601827	

Para estos coeficientes se tiene un valor de RMSE de:

```
In [89]: print(min(listaRMSE))
0.39916021020675224
```

Sin embargo, estos resultados no son constantes y el error tiende a aumentar dependiendo de las acciones realizadas por el algoritmo de ascenso y por los conjuntos de datos.

Bibliografía

- Cathy O'Neil. (2017). Weapons of math destruction. United states: Broadway books.
- Ángel Fernando Kuri Morales. (n.d). Minimax Approximation. México: Instituto Tecnológico Autónomo de México.

Copias electrónicas de los archivos de resultados

Anexos en la carpeta de archivos entregada.