Trabajo Práctico N°1: Ajuste de exponenciales.

Optimización – Primer Cuatrimestre 2017

Maia Numerosky y Matías Zylbersztejn

29 de abril de 2017

1. Introducción

El objetivo de este trabajo es implementar y comparar algoritmos que ajusten conjuntos de datos con exponenciales, siguiendo diferentes metodologías.

A lo largo de todo el texto, asumiremos que los datos vienen dados por vectores $\mathbf{x} = (x_1, \ldots, x_N)$ e $\mathbf{y} = (y_1, \ldots, y_N)$, de manera tal que los puntos (x_i, y_i) caigan aproximadamente en una exponencial. Se consideraron dos clases de funciones exponenciales:

1. Regresión exponencial: $y = k \cdot a^x$
2. Regresión exponencial generalizada: $y = k \cdot e^{b \cdot x} + c$

Una ventaja del primer caso es que es una expresión *intrínsecamente lineal*, o sea que la podemos convertir en una expresión lineal respecto de sus parámetros. Esto permite resolver el problema mediante un modelo de cuadrados mínimos clásico. Sin embargo, como se verifica en este trabajo práctico, los valores obtenidos no necesariamente son los óptimos del problema original. No obstante esta diferencia, veremos que dichos datos nos dan un criterio para elegir un valor inicial para resolver el problema general de manera iterativa mediante el método de Gauss-Newton.

2. Problema 1: Regresión exponencial

Para este problema consideramos primero la siguiente transformación de variables: Sabiendo que $y = k \cdot a^x \xrightarrow{\log} \hat{y} = \hat{k} + x \cdot \hat{a}$. Donde $\hat{y} = \log y$, $\hat{k} = \log k$, y $\hat{a} = \log a$. Esta *linealización* del problema exponencial nos permite resolver el problema de cuadrados mínimos con los métodos lineales clásicos, en los cuales podemos encontrar la solución exacta. Pero como ambos problemas no son equivalentes, las soluciones no tienen por qué coincidir. Pero utilizamos esos valores obtenidos como valores iniciales para resolver iterativamente el problema exponencial original (ver problema 2).

2.1. Resolución

Como vimos, este problema es intrínsecamente lineal. Por lo tanto, en lugar de resolver $\text{mín}_{a,k} \left\| \mathbf{y} - k a^{\mathbf{x}} \right\|$, donde $a^{\mathbf{x}} = \left(a^{x_1}, \ldots, a^{x_n}\right)$, resolvemos $\text{mín}_{a,k} \left\| \hat{\mathbf{y}} - \hat{k} - \hat{a} \cdot \mathbf{x} \right\|$, donde $\hat{\mathbf{y}} = \left(\log y_1, \ldots, \log y_n\right)$. Esto último sí es equivalente a un problema de cuadrados mínimos lineales intentando hacer el mejor ajuste lineal del conjunto de datos $\mathbf{x}$ e $\mathbf{y}$. Por lo tanto, A sería una matriz de dos columnas: una es $(x_1, \ldots, x_N)^t$ y la otra $(1, \ldots, 1)^t$.

Las ecuaciones normales del problema de cuadrados mínimos son

$$A^t \cdot A \cdot \begin{pmatrix} k \\ a \end{pmatrix} = A^t \cdot b \ .$$

Expresando $A = Q \cdot R$ con Q unitaria y R triangular, vemos que esto es equivalente a resolver

$$R \cdot \begin{pmatrix} k \\ a \end{pmatrix} = Q^t \cdot b \ .$$

Y esto no presenta dificultad, de hecho funciona mejor que resolver las ecuaciones normales en el caso de que A esté mal condicionada.

2.1.1. Filtrado de resultados por signo

Evidentemente, es menester que los valores de $\mathbf{y}$ sean positivos para aplicar logaritmo. El problema surge cuando vemos datos de entrada que no lo son, y por lo tanto hay dos opciones: que la mayoría de los datos sean negativos, y por lo tanto estemos en presencia de una exponencial con un $k < 0$; o que decidamos descartar los datos de un determinado signo (usando algún criterio razonable).

Pensamos en tres opciones para resolver este problema:

1. Función `filtraSigno`: Fijarse cuál es el signo mayoritario de $\mathbf{y}$ y eliminar las coordenadas que tienen el signo minoritario. El problema de esto es que, en algunos casos de test, la cantidad de positivos y de negativos era muy similar. Cuando uno observaba el gráfico de los puntos, se notaba que los datos correspondían a una exponencial (por ejemplo) decreciente pero, como teníamos justo la mitad más uno de los datos positivos, quedaba un k positivo. Esto no es solamente un problema que tiene esta manera de encarar el filtrado sino una falla inherente a estar buscando ajustes de forma $\mathbf{y} = \mathbf{k} \cdot \mathbf{a^x}$ cuando la realidad es que hay un término c sumando o restando que "corre" el gráfico hacia arriba o hacia abajo.

2. Función `filtraSigno2`: Fijarse el signo del y_j de mayor módulo y asignárselo a k. El problema obvio con este criterio es que es muy sensible a *outliers*.

3. Función `filtraSigno3`: Una combinación de ambas consiste en tomar el 10 % de los valores mayores en valor absoluto y quedarse con el signo de la mayoría de ellos. Decidimos quedarnos con esta opción por ser más robusta.

De todos modos, somos conscientes de que en la mayoría de los casos uno "mira fijo" el juego de datos que quiere ajustar y en ese momento decide si el ajuste por una exponencial es el más apropiado y el signo de los coeficientes, lo cual suele ser bastante evidente. Esto una computadora no lo puede decidir, todas las funciones de filtrado de signo van a tener casos en los que fallen.

3. Problema 2: Regresión exponencial generalizada

Para este problema en principio no hay una manera evidente para linealizarlo. Por lo que recurrimos directamente al método de Gauss-Newton. La ventaja de considerar a estas familias de funciones exponenciales es que nos permite aproximar de manera adecuada una mayor cantidad de datos, el costo es que hay que estimar un parámetro más.

3.1. Resolución

Si llamamos $d_i(b,k,c) = y_i - k \cdot e^{b \cdot x_i} - c$, el problema de determinar los parámetros b, k, c que mejor ajusten los datos $\mathbf{x}$, $\mathbf{y}$ está dado por:

$$\text{mín}\, F(b,k,c)$$

donde

$$F(b,k,c) = \sum_{i=1}^{N} d_i^2 = \sum_{i=1}^{N} \left(y_i - k \cdot e^{b \cdot x_i} - c\right)^2$$

.

El objetivo es minimizar

$$\sum_{i=1}^{N} d_i(\theta)^2, \quad \theta = (b,k,c).$$

Notamos $\mathbf{d}(\theta) = (d_1(\theta), \ldots, d_N(\theta))$. Idealmente (si los datos coincidieran en una misma exponencial) tendríamos que la solución óptima θ^* satisface $\mathbf{d}(\theta^*) = \mathbf{0}$.

Desarrollando el polinomio de Taylor a orden 1 tenemos que:

$$\mathbf{d}(\theta + \mathbf{h}) \simeq \mathbf{d}(\theta) + J(\theta)\mathbf{h} \sim \mathbf{0},$$

donde J es la matriz diferencial de $\mathbf{d} : \mathbb{R}^3 \to \mathbb{R}^N$ con

$$J(b,k,c) = \left(k \cdot \mathbf{x} \cdot e^{b \cdot \mathbf{x}}, e^{b \cdot \mathbf{x}}, -1\right)$$

El método de Gauss-Newton toma una solución inicial θ_0, resuelve el problema de cuadrados mínimos clásico intentando encontrar el $\mathbf{h}$ tal que se minimice $J(\theta_0)\mathbf{h} + \mathbf{d}(\theta_0)$, toma $\theta_1 = \theta_0 + \mathbf{h}$, e itera el procedimiento, calculando una sucesión θ_i y deteniéndose cuando la función $\mathbf{d}$ se estanca (no tiene por qué tender a cero) o se alcanza una cierta cantidad de iteraciones.

Sabemos que este algoritmo es muy sensible al dato inicial, por eso (tal como estaba sugerido en el enunciado) tomamos como θ_0 el dato de salida del problema 1. Puesto que el mismo se trataba de aproximar los datos por una función del estilo $\mathbf{y} = \mathbf{k} \cdot \mathbf{a}^{\mathbf{x}}$ pero este tiene base e y un coeficiente b, tomamos $\theta_0 = (\log a, k, 0)$.

4. Problema 1bis

Nos pareció que, en cierta manera, los problemas 1 y 2 eran incomparables, ya que estábamos ajustando los datos por funciones distintas (la diferencia es el parámetro c). Desde un comienzo es bastante obvio que el ajuste usando Gauss-Newton será mejor (en términos del error) que linealizando, ya que esto último asume que $c = 0$. Por eso es que probamos a ver qué pasa si intentamos resolver el mismo problema (ajustar con una función del tipo $k \cdot a^x$) usando Gauss-Newton en lugar de linealizando.

5. Resultados y conclusiones

El código del programa fue implementado en MatLab. Se espera como dato de entrada una matriz de N x 2 donde N es la cantidad de pares de datos. Esto se escribe en el código en las líneas `5` y `6`. La primera columna corresponde a la variable independiente, mientras que la segunda a la variable dependiente.

En todos los casos de test subidos a la página de la materia, siempre resultó mejor (en el sentido de que el error siempre fue menor) la resolución por Gauss-Newton con el parámetro c incluido (**Problema 2**), como era de esperar. [1]

En algunos casos, notamos que, después de varias iteraciones, la resolución de **Problema 1bis** "se iba", dando valores muy grandes, del orden de hasta 10^{150} que no tenían nada que ver con los datos ingresados. Sospechamos que eso se debía a errores numéricos, y esto nos impedía muchas veces ver los resultados con claridad. Sin embargo, al ver que muchas veces el error era un número complejo (con parte imaginaria no nula) sospechamos que estábamos haciendo de número negativo el argumento de un logaritmo o la base de una exponencial , y así era: para calcular **J** teníamos que hacer $\log a$ y esto podía ser un número negativo. Por eso es que reemplazamos el problema de ajustar por $k \cdot a^x$ por el equivalente de ajustar con $k \cdot e^{b \cdot x}$. Esto lo resolvimos también por Gauss-Newton y la diferencia es que no hay ningún logaritmo en la matriz diferencial. Esto cambió mucho los gráficos (Ver Figura 1) y nos dimos cuenta de que los de **Problema 1bis** se parecían mucho a los de **Problema 2**, y no así a los de **Problema 1**. Creemos que esto se debe a que en **Problema 1bis** uno no filtra por signo y, por lo tanto, no elimina datos, como sí pasa en **Problema 1**. Por eso (para no ensuciar el código de `main.m`) apartamos el código que intentaba ajustar con $k \cdot a^x$ usando Gauss- Newton en un archivo nuevo llamado `ej1bisfeo.m`. En resumen, en los gráficos subsiguientes se puede observar que:

- El gráfico usando **1bis** (turquesa) se parece bastante al que se obtiene usando **2**, lo cual tiene sentido en el caso de que $c \simeq 0$ (ver Figura 2).
- Cuando hay una cantidad importante de datos positivos y negativos, cumple un rol muy importante el término independiente c. Por eso es que se nota que es muy necesario el ajuste exponencial generalizado, ya que este mejora significativamente el error (ver Figura 3).
- Puede haber errores numéricos graves, como por ejemplo en la Figura 4. Al ver el código, nos dimos cuenta de que se estaba evaluando una expresión muy cercana a "$1 \div 0$"(pues $\left(e^b\right)^x = \frac{1}{(e^b)^{-x}}$), lo cual arroja `inf` en MatLab. Además de eso, la matriz **J** está mal condicionada (entre 10^5 y 10^{77} según el método). Esto puede estar haciendo que el método no converja y que el error no se estabilice sino que tienda a infinito.
- El ajuste fue razonable aún en casos degenerados, ya que se esperaba una recta y eso fue lo que obtuvimos (como se ve en Figura 5).
- Se pusieron a prueba los métodos en datos reales de hospitales que seguían una distribución exponencial y observamos que en este caso real los tres métodos obtuvieron una solución parecida y que ajustaba razonablemente bien los datos (ver Figura 6).

[1] Cada vez que uno quiera testear un juego de datos y corra el programa, el gráfico con menor error tendrá una leyenda de texto rojo.

Figura 1: Distorsión del gráfico de `Datos1` al *plotear* **Problema 1bis**.

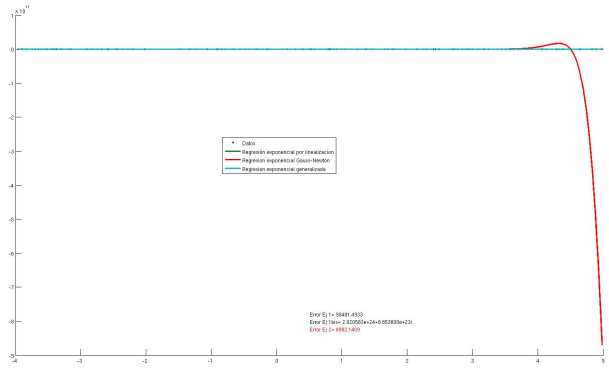


Figura 2: `Datos1`.

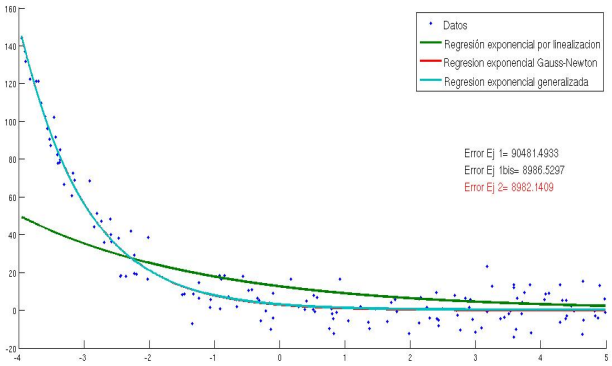


Figura 3: `Datos4`.

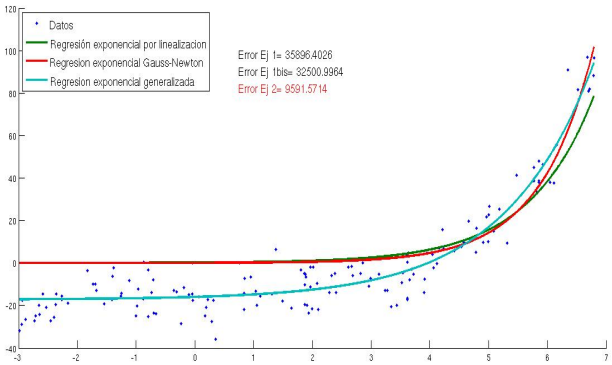


Figura 4: `Datos5`.

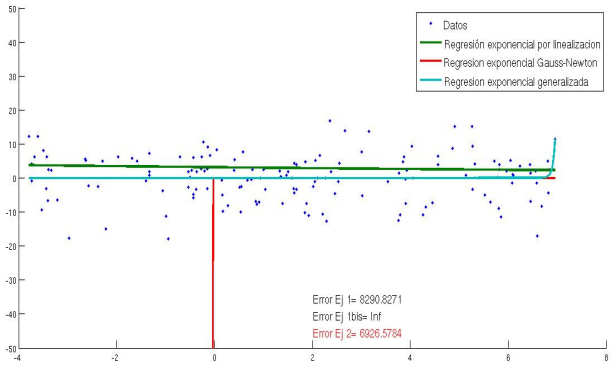


Figura 5: `Datos6`.

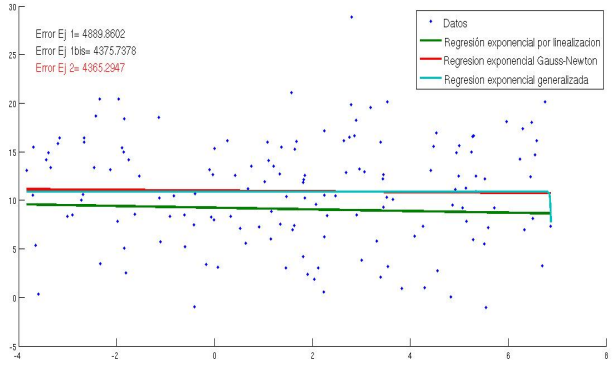
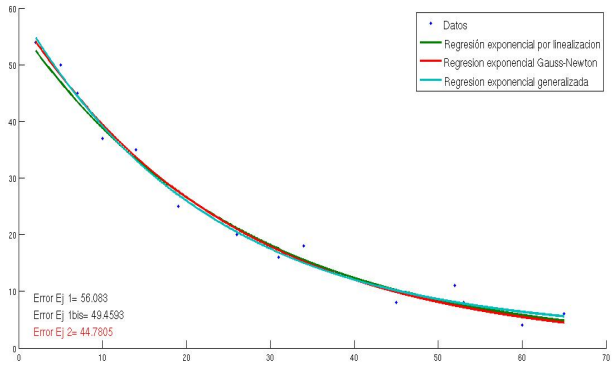


Figura 6: `DatosHospital`.



6. Temas abiertos

Un tema que queda pendiente es un criterio cuantificable de qué tan buena es una regresión exponencial. Pudimos fijar criterios de parada para los métodos iterativos pero los errores variaban mucho entre los datos de entrada.

Algo que no ensayamos fue probar con distintos datos iniciales para el algoritmo de Gauss-Newton. Lo hicimos para `datos5` pero nos seguía ocurriendo el mismo problema de que la función de ajuste se iba a infinito cerca de 0 (por eso es que sospechamos fuertemente que se trata de errores numéricos. Si bien un θ_0 bien elegido puede lograr mejores aproximaciones (dependiendo de los errores numéricos de $\mathbf{J}$), en el caso de que el método converja, lo hace a veces a una función en la que uno termina evaluando algo del estilo "$1 \div 0$ " y eso se vaya a infinito. Para el resto de los juegos de datos, la solución obtenida por linealización parece funcionar razonablemente bien como primer paso para el método de Gauss-Newton, por eso es que no intentamos con otros valores iniciales.

Por último, cabe la posibilidad (para otra instancia) de analizar tiempos de ejecución, complejidad y velocidad de convergencia. Esto no es algo evidente debido a que Gauss-Newton es un método iterativo, mientras que la linealización es un método directo (aquí no cabe el concepto de velocidad de convergencia porque no tenemos una sucesión de datos que queremos que converja). Se puede analizar tiempos de ejecución y costos-beneficios. Uno de los factores a tener en cuenta, por ejemplo, es que en los métodos iterativos uno hace cuadrados mínimos (con lo que implica hacer la descomposición QR, invertir matrices, etc.) muchas veces, pero una sola vez cuando linealiza.