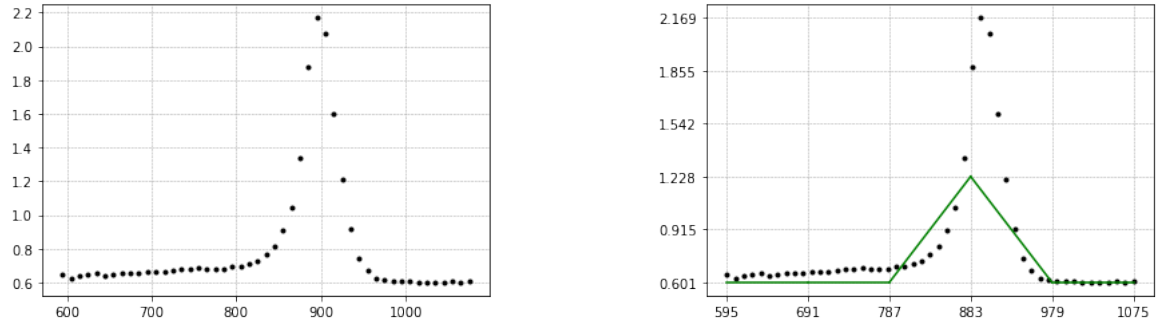
UNIVERSIDAD TORCUATO DI TELLA

Tecnología Digital V: Diseño de algoritmos
Licenciatura en Tecnología Digital
Primer Semestre, 2023

TP1 - Aproximación de datos vía funciones lineales continuas a trozos

Motivación

La aproximación de datos mediante funciones es una herramienta fundamental en diversas disciplinas, ya que permite modelar de manera eficiente fenómenos complejos que exhiben diversos comportamientos. Aplicaciones en ingeniería, química, física, economía y negocios, entre otras, son algunos ejemplos de estas áreas. En muchos casos, los datos se originan o correponden a fenómenos complejos, correspondientes a funciones desconocidas, y con variaciones bruscas o cambios abruptos, con comportamientos intrínsecamente no lineales, no cóncavos ni convexos. Contar con buenas aproximaciones de las funciones subyacentes resulta en un input clave para procesos más complejos.
Las funciones lineales continuas a trozos (PWL, *continuous piecewise linear*) constituyen una herramienta versátil para estos contextos. Intuitivamente, el domino de valores se particiona en una colección de *segmentos*, y dentro de cada segmento los datos se aproximan mediante una función lineal. Para garantizar continuidad, funciones correspondientes a segmentos contiguos deben coincidir en el punto de unión. A modo de ejemplo, la Figura 1a muestra los datos correspondientes a la instancia *titanium*, proveniente de un proceso industrial en base a titanio, y en la Figura 1b los mismos datos junto con una aproximación mediante una función PWL con 4 segmentos. Notar que la aproximación no neceseriamente *interpola* a los puntos.



(a) Datos. (b) Aproximación.

Figure 1: Instancia *titanium*.

La instancia *titanium* cuenta con 49 puntos provenientes de una función $g(t)$, desconocida (Figura 1a). Estos puntos son aproximados mediante una función continua PWL (Figura 1b) con 4 segmentos. A modo de ejemplo, el segundo segmento está definido por la recta que une los puntos (787, 0.601) y (883, 1.228).
En este trabajo práctico vamos a desarrollar métodos para, dado un conjunto de puntos provenientes de una función desconocida, encontrar una función continua PWL que los aproxime minimizando el error. Para ello diseñaremos distintos algoritmos, con distintas implementaciones y realizaremos una evaluación experimental de performance y calidad sobre distintas instancias.

Preliminares

Consideremos una función $g(t)$, desconocida, definida en el intervalo $[a, b]$ de la cual sólo conocemos n puntos $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^2$, con $x_1 < x_2 < \cdots < x_n$, que buscamos aproximar. Dados dos puntos $(t', y'), (t'', y'') \in \mathbb{R}^2$, $t' < t''$ y $[t', t''] \subseteq [a, b]$, la recta que une estos puntos está dada por

$$y(t) = \left[\frac{y'' - y'}{t'' - t'}\right](t - t') + y'. \tag{1}$$

Dado un punto (x_i, y_i) conocido y proveniente de $g(t)$, con $x_i \in [t', t'']$, el error absoluto de aproximación por $y(t)$ (norma L_1, $\|\cdot\|_1$) en x_i está dado por

$$e(x_i, y_i) = |y_i - y(x_i)|.$$

Una función $f(t) : [a, b] \longrightarrow \mathbb{R}$ se dice continua y PWL si existe un número finito K de valores y $K - 1$ funciones lineales $f_k(t)$, con $a = r_1 < r_2 < \cdots < r_K = b$ tal que para $k = 1, \ldots, K - 1$ se cumple que:

- $f(t)$ es una función lineal $f_k(t)$ dentro de cada segmento $[r_k, r_{k+1}]$; y
- $f_k(r_{k+1}) = f_{k+1}(r_{k+1})$.

A cada uno de los puntos r_k se los denomina *breakpoint*, y a cada función $f_k : [r_k, r_{k+1}] \longrightarrow \mathbb{R}$ se la denomina *pieza*.
Para ejemplificar estas definiciones, la instancia *titanium* cuenta con $n = 49$ puntos definidos en el intervalo $[x_1, x_n] = [505, 1075]$ (Figura 1a). Estos puntos son aproximados mediante una función continua PWL $f(x)$ compuesta de 4 piezas y 5 breakpoints. Para el primer segmento, $r_1 = 595$, $r_2 = 787$, y la pieza $f_1(t)$ se obtiene mediante la función lineal que une los puntos (595,0.601) y (787,0.601), siguiendo la ecuación (1). Análogamente, la pieza $f_2(t)$ tiene dominio $[r_2, r_3] = [787, 883]$ y la función $f_2(t)$ se obtiene aplicando la ecuación (1) tomando como referencia los puntos $(787, 0.601)$ y $(883, 1.228)$. Notar que una función continua PWL puede ser definida en términos de K puntos dados por $(r_k, f_k(r_k))$ para $k = 1, \ldots, K - 1$ y $(r_K, f_{K-1}(r_K))$.
Finalmente, analizamos el error de la aproximación. Dada una pieza $f_k(t)$ definida por los breakpoints (r_k, z_k) y (r_{k+1}, z_{k+1}) y los puntos (x_i, y_i), $i = 1, \ldots, n$, definimos el error de aproximación de la pieza k-ésima como la suma de los errores de los puntos (x_i, y_i) tal que $x_i \in (r_k, r_{k+1}]$, es decir,

$$D((r_k, z_k), (r_{k+1}, z_{k+1})) = \sum_{r_k < x_i \leq r_{k+1}} |y_i - f_k(x_i)|.$$

A fin de considerar todos los puntos, para $f_1(t)$ la sumatoria debe considerar el intervalo cerrado $[r_1, r_2]$ (i.e., en caso que $r_1 = x_1$). Dada una función PWL $f_(t)$, el error de aproximación de la misma está dado por la suma de los errores de aproximacíon de cada una de sus piezas.

El problema

Sean $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^2$ una serie n puntos, ordenados de forma que $x_1 < x_2 < \cdots < x_n$, provenientes de una función $g(t)$ desconocida que buscamos aproximar. Consideramos también una grilla correspondiente a la discretización del área $[x_1, x_n] \times [y_1, y_n] \subset \mathbb{R}^2$. Dado el rango $[x_1, x_n]$ para los valores de la abscisa, consideramos una discretización en m_1 puntos $x_1 = t_1 < t_2 < \cdots < t_{m_1} = x_n$ con $t_i = t_1 + (i - 1) \times \Delta x$, con $\Delta x = (x_n - x_1)/(m_1 - 1)$.

Análogamente, el rango $[y_1, y_n]$ para los valores de la ordenada es discretizado en m_2 puntos $y_1 = z_1 < z_2 < \cdots < z_{m_2} = y_n$ con $z_j = z_1 + (j-1) \times \Delta y$, con $\Delta y = (y_n - y_1)/(m_2 - 1)$. Siguiendo con el ejemplo de la instancia *titanium*, el rango $[x_1, x_n] = [595, 1075]$ está discretizado tomando $m_1 = 6$, $\Delta x = 96$ y los puntos resultantes son $[595, 691, 787, 893, 979, 1075]$. De forma similar, el rango $[y_1, y_2] = [0.601, 2.169]$ también se encuentra discretizado tomando $m_2 = 6$ puntos.
El objetivo del trabajo práctico consiste tomar como input los valores $(x_i, y_i), i = 1, \ldots, n$, una discretización (t_i, z_j) con $i = 1, \ldots, m_1, j = 1, \ldots, m_2$, un valor K para la cantidad de breakpoints y cacular una función continua PWL $f(t) : [x_1, x_n] \longrightarrow \mathbb{R}$ con K *breakpoints* ($K-1$ piezas $f_1(t), \ldots, f_{K-1}(t)$) pertenecientes a la discretización (t_i, z_j) y que minimicen el error total.

Enfoque basado en Programación Dinámica

En [1] se propone un algoritmo basado en la técnica de programación dinámica para el problema propuesto, que describimos a continuación. Sea $F_K((t_i, z_j))$ el mínimo error de aproximar la curva $g(t)$ usando exactamente K piezas cuando el último punto de la última pieza es el punto (t_i, z_j).
Analizamos primero la estructura del problema. La solución óptima estará dada por el mínimo de los valores que se encuentran en la última columna de la discretización, es decir, $F_K(t_{m_1}, z_j)$, para algún $j = 1, \ldots, m_2$. En el ejemplo de la instancia *titanium*, asumiendo que la función provista es óptima, corresponde al punto $(1075, 0.601)$. Si tenemos una función cuya última pieza $f_K(t)$ está definida por los breakpoints (t_k, z_l) y (t_{m_1}, z_j)[1], entonces el error mínimo para una función continua PWL que utilice necesariamente esta pieza está dado por

- el error cometido por la aproximación de la pieza (t_k, z_l) y (t_{m_1}, z_j), y
- el mínimo error cometido de todas las funciones continuas PWL de $K-1$ piezas cuyo último breakpoint sea (t_k, z_l), es decir, que se *pegan* con el la última pieza en (t_k, z_l) (para garantizar continuidad).

De esta forma, el valor óptimo puede calcularse a partir de soluciones óptimas de subproblemas, mostrando una *subestructura óptima*. Generalizando esta idea para cualquier punto de la grilla, para $M > 1$ podemos plantear la siguiente forma funcional recursiva

$$F_M((t_i, z_j)) = \min_{\substack{(t_k, z_l): t_1 < t_k < t_i \\ l = 1, \ldots, m_2}} \Big\{ D((t_k, z_l), (t_i, z_j)) + F_{M-1}((t_k, z_l)) \Big\}, \tag{2}$$

con el correspondiente caso base (con $i > 1$)

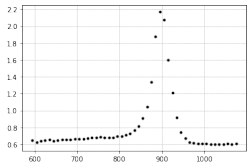
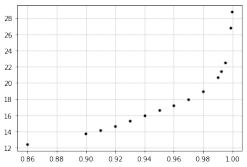
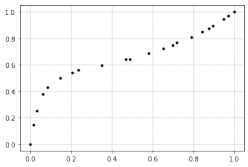
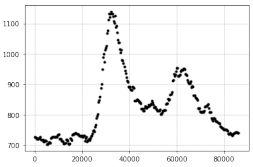
$$F_1((t_i, z_j)) = \min_{\substack{(t_1, z_l) \\ l = 1, \ldots, m_2}} D((t_1, z_l), (t_i, z_j)). \tag{3}$$

Una vez calculados todos estos valores, es posible reconstruir la solución partiendo del valor óptimo (t_{m_1}, z_{j_0}), con $z_{j_0} = \arg\min_{j=1,\ldots,m_2} F(t_{m_1}, z_j)$.

Los datos

Para la realización del trabajo se contará con 4 conjuntos de datos correspondientes a los valores $(x_1, y_1), \ldots, (x_n, y_n)$, provenientes de aplicaciones de distintos dominios (Figura 2).

[1] Notar que el úlitmo breakpoint está definido en la última columna de la grilla de la discretización, $i = m_1$.

(a) *titanium*, $n = 49$ (b) *aspen*, $n = 15$ (c) *ethanol*, $n = 22$ (d) *google*, $n = 288$

Figure 2: Instancias.

Los datos se proveen en formato JSON, con los siguientes campos:

- `n`: cantidad de puntos (`int`).
- `x`: lista de `double` con los valores $x_1, \ldots, x_n$.
- `y`: lista de `double` con los valores $y_1, \ldots, y_n$.

Se remarca que tanto las discretizaciones y el valor K para el número de piezas, que también son necesarias para definir una instancia del problema, no forman parte del input y serán un parámetro de experimentación a definir por el grupo. Junto con este enunciado se incluyen algunos templates básicos para levantar y visualizar instancias y soluciones, que pueden utilizar durante el desarrollo en caso de ser conveniente.

El trabajo

El trabajo práctico debe ser realizado en grupos de exactamente 3 personas. La resolución del trabajo se compone de varias etapas, incluyendo el diseño de 3 algoritmos, programación, experimentación, así también como el reporte detallado de la evaluación de los métodos implementados, los resultados obtenidos y la discusión sobre el caso real. **Los algoritmos deben ser implementados en `Python` y `C++`**. Se pide:

1. **(1.5 puntos) Fuerza bruta.** Proponer e implementar un algoritmo de fuerza bruta.
2. **(1.0 puntos) Backtracking.** Partiendo de la implementación anterior, implementar un backtracking incorporando al menos 1 poda por factibilidad y 1 poda por optimalidad.
3. **(2.0 puntos) Programación dinámica.** Implementar el algoritmo de programación dinámica descripto anteriormente, incluyendo la rutina de reconstrucción de la solución
4. **(2.0 puntos) Experimentación y discusión.** Realizar una experimentación exhaustiva usando (al menos) los datasets provistos. Considerando distintos valores posibles para la discretización (m_1 y m_2) así como también para la cantidad de piezas a considerar (K breakpoints), abordar las siguientes preguntas:
    - *Calidad*: ¿Cómo impacta la granularidad de la discretización y la cantidad de piezas seleccionadas en la calidad de las aproximaciones?
    - *Performance*: ¿Cuáles son los paŕametros de las instancias que mayor impacto tienen en la performance de los algoritmos? ¿Cómo afecta la elección del lenguaje de programación y la implementación elegida?
    - *Propuesta*: En base a los análisis previos, ¿cuál sería la sugerencia del grupo respecto al algoritmo, implementación y lenguaje de programación?

5. **(3.5 puntos) Algoritmos, informe, presentación de resultados y delivery del código.** El modelo, la descripción de los algoritmos, los operadores, las decisiones de diseño, la implementación, el testing realizado, la presentación de resultados, instrucciones de compilación y ejecución.

Modalidad de entrega

Se pide presentar el modelo y la experimentación en un informe de máximo 15 páginas que contenga:

- introducción al problema y la decisión,
- descripción de los algoritmos propuestos e implementados, según corresponda,
- consideraciones generales respecto a la implementación del modelo, incluyendo dificultades que hayan encontrado,
- resumen de resultados obtenidos en la experimentación,
- conclusiones, posibles mejoras y observaciones adicionales que consideren pertinentes.

Junto con el informe debe entregarse el código con la implemenetación del modelo. El mismo debe ser entendible, incluyendo comentarios que faciliten su corrección y ejecución.

Fechas de entrega

Formato Electrónico: **viernes 19 de abril de 2024, 23:59 hs**, enviando el trabajo (informe + código) vía el campus virtual.
Importante: El horario es estricto. Las entregas recibidas después de la hora indicada serán consideradas re-entrega.

References

[1] Richard Bellman and Robert Roth. Curve fitting by segmented straight lines. *Journal of the American Statistical Association*, 64(327):1079–1084, 1969.