02MIAR
Actividad evaluada — ejercicios

Dr. Matthieu F.-W. Huber

Junio de 2024

Instrucciones:

- resolver los ejercicios rigurosamente y con buen fundamento,
- entregar su resolución tecleada a máquina (se puede usar TEX),
- incluir el código fuente del módulo Python en el que aparezca su implementación de los algoritmos de cada ejercicio.

Plazo de entrega: el día 21 de junio de 2024 a las 23:59.

1. Ejercicios acerca del determinante

Ejercicio 1 (2 puntos)**.** *Desarrollo de Laplace.*

1. *Deducir de la definición 4 el determinante en dimensión* 0*,* 1 *y* 2*.*

2. *A partir de la definición 4, expresar el determinante de una matriz cuadrada recursivamente en función de determinantes la matrices cuadradas de dimensión inferior.*

   Indicación: para cada $n \in \mathbb{N}$*, distribuir (por linealidad en las columnas) sobre la descomposición*

   $$\begin{bmatrix} \lambda & \omega \\ v & A \end{bmatrix} = \begin{bmatrix} \lambda \cdot 1 + 0 & \omega \\ \lambda \cdot 0 + v & A \end{bmatrix}$$

   de una matriz cuadrada de dimensión $n + 1$*, siendo* $n \in \mathbb{N}$ *y*

   - λ *un coeficiente real,*
   - v *un vector de dimensión* n *(una columna de* n *coeficientes reales),*
   - ω *un covector de la misma dimensión (una fila de* n *coeficientes),*
   - A *una matriz cuadrada de la misma dimensión (con* n^2 *coeficientes),*

   luego proceder del mismo modo

   - *con los demás coeficientes de esa primera columna,*
   - *con cada columna.*

3. *Implementar en Python la definición así obtenida.*

Ejercicio 2 (2 puntos)**.** *Eliminación de Gauss–Jordan.*

1. *Deducir de la definición 4 el efecto que tiene en el determinante de una matriz sumar a una de sus columnas una combinación lineal de las demás.*

2. *A partir de la definición 4, proponer una estrategia para triangularizar una matriz sin cambiar su determinante e implementar en Python una definición alternativa del determinante.*

   Indicación: descomponer similarmente al ejercicio anterior.

3. *implementar en Python la definición así obtenida.*

Ejercicio 3 (2 puntos)**.** *Comparación.*

1. *Obtener la complejidad computacional de cada una de estas dos implementaciones.*

2. *Generar matrices aleatoriamente en dimensión* $n \in \{2, 3, \cdots, 9, 10\}$ *y comparar el tiempo de ejecución de cada una de estas dos implementaciones con la función* `numpy.linalg.det` *(la función determinante de la extensión numérica de Python al álgebra lineal).*

   Indicación: se puede utilizar la función `numpy.random.rand` *para generar los coeficientes aleatorios de sus matrices.*

2. Ejercicio acerca del gradiente

Ejercicio 4 (4 puntos)**.** *Con el propósito de aproximar un mínimo local de una función real de varias variables reales, el* método de descenso de gradiente *consiste en iterar una marcha (positivamente) proporcional al (opuesto del) gradiente desde un valor inicial, con la intuición de 'seguir el agua' hasta dar con el valle.*

1. *Implementar en Python un algoritmo de descenso de gradiente (con un máximo de* `m = 10**5` *iteraciones) a partir de los siguientes argumentos tomados en ese orden:*

   - *la función* `f` *cuyo mínimo local se propone aproximar,*
   - *el valor inicial* `x` *desde el que empieza la marcha,*
   - *la razón geométrica o coeficiente de proporcionalidad* `y`*,*
   - *el parámetro de tolerancia* `z` *para finalizar cuando el gradiente de la función* `f` *caiga dentro de esa tolerancia.*

   Indicación: empezar por implementar el gradiente `grad(f)` *de la función* `f`*.*

2. *Calcular formalmente* $\{ t \in \mathbb{R}. \quad f'(t) = 0 \}$ *para* $f : t \mapsto 3t^4+4t^3-12t^2+7$*.*

3. *Con una tolerancia* `z = 10**-12` *y un valor inicial de* `x = 3` *aplicar su algoritmo con razón* `y = 10**-1, 10**-2, 10**-3` *luego hacer lo mismo con* `x = 0`*. Interpretar el resultado.*

4. *Repetir estos dos últimos apartados con* $f : (s, t) \mapsto s^2 + 3st + t^3 + 1$ *y los valores iniciales* `x = [-1,1], [0,0]`*.*

3. Anexo – definiciones fundamentales

Definition 1 (Espacio euclidiano)**.** *El* espacio euclidiano *es el conjunto de las tuplas de unos coeficientes reales (su* dimensión *es el número de coeficientes), equipado de la* suma vectorial *coeficiente por coeficiente y de la* multiplicación por un escalar *(real) en todos los coeficientes a la vez.*

Definition 2 (Transformación lineal)**.** *Una función con entrada y salida en espacios euclidianos es* lineal *cuando es distributiva sobre la suma coeficiente a coeficiente y conmuta con la multiplicación por un escalar. La* matriz *de una función que sea una transformación lineal es la lista de los vectores a los que son llevados los vectores de la base canónica (formado por las tuplas cuyos coeficientes son todos cero salvo un coeficiente igual a uno).*

Definition 3 (Grupo lineal general)**.** *Una función con entrada y salida en el mismo espacio euclidiano es invertible cuando admite otra función tal que su composición funcional con ella sea la función identidad de ese espacio euclidiano. Esas dos funciones serán cada una la inversa de la otra. El* grupo lineal general *del espacio euclidiano (de una dimensión dada) es el conjunto de aquellas transformaciones invertibles de ese espacio que son lineales, equipado de la composición funcional.*

Definition 4 (Determinante)**.** *El* determinante *de una matriz en tanto lista de vectores es la única función lineal antisimétrica de estos vectores tal que la matriz de la función identidad tenga determinante uno.*

4. Breve recordatorio – matrices en Python

Una lista se especifica en Python por medio de unos paréntesis cuadrados, explicitando sus ítems, por ejemplo

```
x = [ 1, 4, 9 ]
```

o bien por comprensión de listas, en nuestro ejemplo

```
y = [ (n+1)**2 for n in range(0,3) ]
```

evaluará `x == y` como `True`. Cada ítem se localiza en una lista especificando su índice, por ejemplo

```
z = [ x[i] for i in range(0,len(x)) ]
```

evaluará `z == x` como `True` para cualquier lista `x`. Si una columna

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = a \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + c \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^3$$

de tres coeficientes reales $a, b, c \in \mathbb{R}$ representa un vector del espacio euclidiano $\mathbb{R}^3$ de dimensión tres, su imagen

$$f\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}\right) = af\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) + bf\left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) + cf\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) \in \mathbb{R}^n$$

por cualquier función lineal $f : \mathbb{R}^3 \to \mathbb{R}^n$ dependerá solamente del valor

$$f\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right), \quad f\left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right), \quad f\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right)$$

que toma esa función en los vectores

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

de la base canónica del espacio $\mathbb{R}^3$. Introducimos la matriz de f como fila

$$\begin{bmatrix} f\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) & f\left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) & f\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) \end{bmatrix}$$

de estos valores (coeficientes individuales si $n = 1$, columnas de varios coeficientes para $n \geq 2$). En particular, representaríamos el covector

$$\begin{array}{rccl} & \mathbb{R}^3 & \to & \mathbb{R} \\ \begin{bmatrix} a \\ b \\ c \end{bmatrix}^* : & \begin{bmatrix} x \\ y \\ z \end{bmatrix} & \mapsto & ax + by + cz \end{array}$$

asociado al vector

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \in \mathbb{R}^3$$

por la fila

$$\begin{bmatrix} a & b & c \end{bmatrix} = \Big[\underbrace{1a+0b+0c}_{\begin{bmatrix} a \\ b \\ c \end{bmatrix}^* \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}} \quad \underbrace{0a+1b+0c}_{\begin{bmatrix} a \\ b \\ c \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}} \quad \underbrace{0a+0b+1c}_{\begin{bmatrix} a \\ b \\ c \end{bmatrix}^* \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}} \Big] \in (\mathbb{R}^3)^*$$

que en Python sería una lista de listas. Por ejemplo, el vector

```
v = [ [1,4,9] ]
```

(en tanto matriz de una sola columna) tendría covector

```
w = [ [1], [4], [9] ]
```

(en tanto matriz de una fila o, de manera equivalente, cuyas columnas comportan cada una un solo coeficiente) con acción dada por el producto

```
lambda w, v: sum([ w[i][0]*v[0][i]
                      for i in range(0,len(v)) ])
                if len(v) == len(w)
                else None
```

que se generaliza al producto de matrices

$$A \underbrace{\begin{bmatrix} u & v & \cdots & w \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} Au & Av & \cdots & Aw \end{bmatrix}}_{AB}$$

en tanto matriz de la composición funcional de las funciones lineales que representan, esto es,

```
lambda n, A, B: [ [ sum([ A[i][j]*B[k][i]
                             for i in range(0,n) ])
                       for j in range(0,len(A[0])) ]
                  for k in range(0,len(B)) ]
```

salvo condición de compatibilidad (que el número de columnas en la primera matriz A coincida con el número de coeficientes en cada columna de la segunda matriz B). Ese producto no conmuta en general: por ejemplo, el producto de la matriz

```
A = [ [1,0], [0,1], [2,2] ]
```

que representa (con nuestra convención) la función lineal

$$\begin{aligned} \mathbb{R}^3 &\to \mathbb{R}^2 \\ \begin{bmatrix} x \\ y \\ z \end{bmatrix} &\mapsto \begin{bmatrix} x+2z \\ y+2z \end{bmatrix} \end{aligned}$$

con la matriz

```
B = [ [3,3,3], [1,0,-1] ]
```

que representa la función lineal

$$\begin{aligned} \mathbb{R}^2 &\to \mathbb{R}^3 \\ \begin{bmatrix} x \\ y \end{bmatrix} &\mapsto \begin{bmatrix} 3x+y \\ 3x \\ 3x-y \end{bmatrix} \end{aligned}$$

será

```
AB = [ [9,9], [-1,-2] ]
```

en el orden de composición $\mathbb{R}^2 \to \mathbb{R}^3 \to \mathbb{R}^2$ y

```
BA = [ [3,3,3], [1,0,-1], [8,6,4] ]
```

en el otro orden $\mathbb{R}^3 \to \mathbb{R}^2 \to \mathbb{R}^3$.

Nota bene: se puede perfectamente cambiar la convención para que los vectores sean listas de listas de un solo coeficiente, en vez de los covectores que en cuyo caso se convertirán en listas de una sola lista de coeficientes. En el formalismo algebraico, corresponde al intercambio de filas y columnas (transposición) junto a la conmutación del orden de multiplicación de matrices que conlleva.

El propósito de las *extensiones numéricas de Python* (`NumPy`) es aliviar la carga cognitiva de la implementación de estos detalles fundamentales del álgebra lineal en unas funciones predefinidas.

5. Breve recordatorio – gradiente de una función

A partir de unas funciones de coordenadas locales x^i que asignan a cada punto de un cierto espacio (una variedad diferencable) sus respectivos coeficientes en el espacio euclidiano (en tanto espacio de coordenadas cartesianas), se caracteriza el espacio tangente en cada punto como el espacio de las derivaciones v que llevan cada función real diferenciable f a la derivada direccional $v(f)$ en ese punto. Ese espacio tangente es un espacio vectorial con respecto a la suma $(v + w)(f) = v(f) + w(f)$ elemento por elemento y a la multiplicación $(\lambda v)(f) = \lambda\, v(f)$ por un escalar (real) λ homogenea en cada elemento. Haciendo lo mismo en todos los puntos a la vez, se define el espacio de los campos vectoriales como el espacio de las derivaciones V que llevan cada función real diferenciable f a la función real $p \mapsto v(f)$, donde v es el valor (vectorial) que toma el campo vectorial en el punto p, tal que

$$V(fh) = V(f)h + fV(h)$$

siendo f, h funciones reales diferenciables y $fh\colon p \mapsto f(p)h(p)$ su producto punto por punto.

Para cada aplicación diferenciable φ entre dos espacios equipados con funciones de coordenadas locales, se define la diferencial $\mathrm{d}\varphi$ de esa aplicación como la procesión o *'push forward'* de cada vector tangente por retrocesión o *'pull back'* de las funciones reales diferenciables en las que se caracteriza ese vector tangente por su acción en tanto derivación. Esto es,

$$(\mathrm{d}\varphi \cdot V)(h) = V(h \circ \varphi) \tag{1}$$

para cada campo vectorial V y cada función real diferenciable h, siendo $h \circ \varphi\colon p \mapsto f\big(\varphi(p)\big)$ su composición funcional con la aplicación φ.

Se deduce con facilidad que la diferencial es lineal en el sentido de que es distributiva sobre la suma de campos vectoriales punto por punto, $\mathrm{d}\varphi\cdot(V+W) = \mathrm{d}\varphi\cdot V + \mathrm{d}\varphi W$, y homogénea con respecto a la multiplicación por un campo escalar, $\mathrm{d}\varphi \cdot (hV) = h\mathrm{d}\varphi \cdot V$. En particular, la diferencial $\mathrm{d}f$ de cualquier función real diferenciable f será en cada punto un covector cuyos coeficientes en coordenadas locales son las imágenes $\mathrm{d}f \cdot e_i = e_i(f)$ de los vectores e_i de la base inducida en ese punto por las funciones de coordenadas locales. También se obtiene sencillamente que las diferenciales $\mathrm{d}x^i$ de esas funciones de coordenadas locales forman en cada punto una

base del espacio de los covectores, de manera que $\mathrm{d}f = f_i \mathrm{d}x^i$ (con suma implícita) para unas funciones reales f_i coeficientes de la diferencial de la función real diferenciable f. Por la definición (1) con $\varphi = f$, se recupera el hecho ya conocido en el contexto del análisis funcional real de varias variables reales que estos coeficientes son las derivadas direccionales $f_i = \frac{\partial f}{\partial x^i}$, pero ahora con la generalización de esta noción a la geometría diferencial escribiremos

$$V(f) = \mathrm{d}f \cdot V = \frac{\partial f}{\partial x^i} \underbrace{\mathrm{d}x^i \cdot V}_{V^i} = V^i \frac{\partial f}{\partial x^i} = V^i \frac{\partial}{\partial x^i}(f)$$

(con suma implícita) lo cual justifica escribir $V = V^i \frac{\partial}{\partial x^i}$ designando por $\frac{\partial}{\partial x^i}$ el campo vectorial que en cada punto se evalúa en el vector de base e_i.

En el espacio euclidiano de dimensión $n \in \mathbb{N}$, las proyecciones

$$\begin{array}{rcl} \mathbb{R}^n & \to & \mathbb{R} \\ x^i : \begin{bmatrix} t_0 \\ \vdots \\ t_i \\ \vdots \\ t_{n-1} \end{bmatrix} & \mapsto & t_i \end{array}$$

de cada tupla en cada una de sus n coeficientes forman unas funciones de coordenadas locales con respecto a las cuales la definición geométrica de la diferencial $\mathrm{d}f$ de una función real diferenciable $f : \mathbb{R}^n \to \mathbb{R}$ coincide con su expresión analítica

$$\begin{bmatrix} \frac{\partial f}{\partial x^0} & \cdots & \frac{\partial f}{\partial x^i} & \cdots & \frac{\partial f}{\partial x^{n-1}} \end{bmatrix}$$

(definida para f en tanto función real de varias variables reales) que en cada punto consiste en un covector $\omega : \mathbb{R}^n \to \mathbb{R}$ cuyo dual (el vector v tal que $\langle v, w \rangle = \omega \cdot w$ para cada $w \in \mathbb{R}^n$) es el *gradiente*

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x^0} \\ \vdots \\ \frac{\partial f}{\partial x^i} \\ \vdots \\ \frac{\partial f}{\partial x^{n-1}} \end{bmatrix}.$$