Programación: ortogonalización con el algoritmo modificado de Gram–Schmidt

Objetivos. Aplicar el algoritmo modificado de Gram–Schmidt a cuatro vectores, luego generalizar la idea al caso de m vectores.

Requisitos. Proyección ortogonal de un vector al subespacio generado por un vector no nulo, estructura del producto de matrices, ciclos de tipo `for`.

Proyección ortogonal de un vector al subespacio generado por un vector no nulo (repaso)

1. Proyección ortogonal sobre el subespacio generado por un vector normalizado (repaso). Sea $a \in \mathbb{R}^n$ con $\|a\|_2 = 1$. Escriba la fórmula para la matriz de proyección ortogonal sobre el subespacio generado por a:

$$P_a =$$

Calcule la transpuesta de la matriz P_a, el cuadrado de la matriz P_a y el vector $P_a a$:

$$P_a^\top = \qquad\qquad P_a^2 =$$

$$P_a a =$$

2. Cálculo de la proyección ortogonal sobre el subespacio generado por un vector normalizado (repaso). Si $v \in \mathbb{R}^n$, entonces el producto $P_a v$ se puede escribir de varias maneras equivalentes; la forma más útil es

$$P_a v = \qquad (\qquad v).$$

3. Proyección ortogonal simultanea de varios vectores sobre el subespacio generado por un vector (repaso). Sea $a \in \mathbb{R}^n$ con $\|a\|_2 = 1$ y sea $V \in \mathcal{M}_{n\times m}(\mathbb{R})$. Usemos la siguientes notación:

- $v_1, \ldots, v_m$ son las columnas de V, esto es, $v_k = V_{*,k}$;
- $\lambda_1, \ldots, \lambda_m$ son los productos internos de a con $v_1, \ldots, v_m$: $\lambda_k = \langle a, v_k \rangle = a^\top v_k$.
- Λ es el renglón formado por $\lambda_1, \ldots, \lambda_m$: $\Lambda = \big[\lambda_1, \ldots, \lambda_m\big]$.

Exprese en estos términos el producto $a^\top V$:

Encuentre una fórmula para calcular la matriz formada por las columnas $P_a v_1, \ldots, P_a v_m$:

$$\big[P_a v_1, \ldots, P_a v_m\big] = \qquad\qquad = \underbrace{\qquad}_{?} \; \underbrace{\qquad V}_{\Lambda}.$$

4. Función que calcula la proyección simultanea de varios vectores sobre el subespacio generado por un vector (repaso).

```
function [U] = orthproj(a, V),
   Lambda =
   U =
end
```

5. Propiedades de la proyección complementaria (repaso). Sea $a \in \mathbb{R}^n$, $\|a\|_2 = 1$. Definimos P_a como en los ejercicios anteriores. Consideremos la matriz

$$Q_a \coloneqq I_n - P_a.$$

Usando la propiedad $P_a^\top = P_a$ muestre que Q_a es simétrica:

$$Q_a^\top =$$

Usando la propiedad $P_a^2 = P_a$ calcule Q_a^2:

$$Q_a^2 = \qquad\qquad = \underbrace{\qquad}_{?}.$$

Calcule $Q_a a$:

$$Q_a a =$$

Muestre que si $v \in \mathbb{R}^n$ y $w = Q_a v$, entonces $a \perp w$:

$$\langle a, w\rangle = \langle a, Q_a v\rangle = a^\top Q_a v = a^\top Q_a^\top v = \big(\qquad\qquad\big)^\top v =$$

Muestre que si $v \in \mathbb{R}^n$ y $w = Q_a v$, entonces $w \in \ell(a, v)$ y $v \in \ell(a, w)$:

$$w = \qquad\qquad \in \ell(a, v);$$

$$v = \qquad\qquad \in \ell(a, w).$$

Decimos que w es el complemento ortogonal de v respecto a.

6. Cálculo de la proyección ortogonal complementaria (repaso). En la siguiente función se supone que la matriz V tiene n renglones, donde n es la longitud del vector a. La función debe regresar la matriz $W = Q_a V$.

```
function [W] = orthcomplement(a, V),
   Lambda =
   W =
end
```

Algoritmo modificado de Gram–Schmidt para 4 vectores

7. Fórmulas matemáticas. Sean $a_1, a_2, a_3, a_4 \in \mathbb{R}^n$ algunos vectores linealmente independientes. Los guardamos en $b_1^{(0)}, b_2^{(0)}, b_3^{(0)}, b_4^{(0)}$.

En el paso 1 calculamos $b_1^{(1)}$ como $b_1^{(0)}$ normalizado, luego calculamos $b_2^{(1)}, b_3^{(1)}, b_4^{(1)}$ como los complementos ortogonales de $b_2^{(0)}, b_3^{(0)}, b_4^{(0)}$ respecto al vector $b_1^{(1)}$:

$$\nu_1 \coloneqq \|b_1^{(0)}\|_2,$$

$$b_1^{(1)} \coloneqq \qquad /\nu_1,$$

$$\lambda_{1,2} \coloneqq (b_1^{(1)})^\top \underbrace{\qquad}_{?}, \qquad \lambda_{1,3} \coloneqq (b_1^{(1)})^\top \underbrace{\qquad}_{?}, \qquad \lambda_{1,4} \coloneqq (b_1^{(1)})^\top \underbrace{\qquad}_{?},$$

$$b_2^{(1)} \coloneqq b_2^{(0)} - b_1^{(1)} \underbrace{\qquad}_{?}, \qquad b_3^{(1)} \coloneqq b_3^{(0)} - b_1^{(1)} \underbrace{\qquad}_{?}, \qquad b_4^{(1)} \coloneqq b_4^{(0)} - b_1^{(1)} \underbrace{\qquad}_{?}.$$

En el paso 2 calculamos $b_2^{(2)}$ como $b_2^{(1)}$ normalizado, luego calculamos $b_3^{(2)}, b_4^{(2)}$ como los complementos ortogonales de $b_3^{(1)}, b_4^{(1)}$ respecto al vector $b_2^{(2)}$:

$$\nu_2 \coloneqq$$

$$b_2^{(2)} \coloneqq$$

$$\lambda_{2,3} \coloneqq \qquad\qquad \lambda_{2,4} \coloneqq$$

$$b_3^{(2)} \coloneqq \qquad\qquad b_4^{(2)} \coloneqq$$

En el paso 3 calculamos $b_3^{(3)}$ como $b_3^{(2)}$ normalizado, luego calculamos $b_4^{(3)}$ como el complemento ortogonal de $b_4^{(2)}$ respecto al vector $b_3^{(3)}$:

$$\nu_3 \coloneqq$$

$$b_3^{(3)} \coloneqq$$

$$\lambda_{3,4} \coloneqq$$

$$b_4^{(3)} \coloneqq$$

En el paso 4 calculamos $b_4^{(4)}$ como $b_4^{(3)}$ normalizado:

$$\nu_4 \coloneqq$$

$$b_4^{(4)} \coloneqq$$

8. Fórmulas matemáticas más compactas. Sean $a_1, a_2, a_3, a_4 \in \mathbb{R}^n$ algunos vectores linealmente independientes. Los guardamos en $b_1^{(0)}, b_2^{(0)}, b_3^{(0)}, b_4^{(0)}$.

En el paso $p = 1$ calculamos $b_1^{(1)}$ como $b_1^{(0)}$ normalizado, luego calculamos $b_2^{(1)}, b_3^{(1)}, b_4^{(1)}$ como los complementos ortogonales de $b_2^{(0)}, b_3(0), b_4^{(0)}$ respecto al vector $b_1^{(1)}$:

$$\begin{aligned}
\nu_1 &:= \\
b_1^{(1)} &:= \\
\big[\lambda_{1,2}, \lambda_{1,3}, \lambda_{1,4}\big] &:= (b_1^{(1)})^\top \big[\quad , \quad , \quad\big]; \\
\big[b_2^{(1)}, b_3^{(1)}, b_4^{(1)}\big] &:= \big[b_2^{(0)}, b_3^{(0)}, b_4^{(0)}\big] - b_1^{(1)} \big[\underbrace{\quad}_{?}, \underbrace{\quad}_{?}, \underbrace{\quad}_{?}\big].
\end{aligned}$$

En el paso $p = 2$ calculamos $b_2^{(2)}$ como $b_2^{(1)}$ normalizado, luego calculamos $b_3^{(2)}, b_4^{(2)}$ como los complementos ortogonales de $b_3^{(1)}, b_4^{(1)}$ respecto al vector $b_2^{(2)}$:

$$\begin{aligned}
\nu_2 &:= \\
b_2^{(2)} &:= \\
\big[\lambda_{2,3}, \lambda_{2,4}\big] &:= \\
\big[b_3^{(2)}, b_4^{(2)}\big] &:=
\end{aligned}$$

En el paso $p = 3$ calculamos $b_3^{(3)}$ como $b_3^{(2)}$ normalizado, luego calculamos $b_4^{(3)}$ como el complemento ortogonal de $b_4^{(2)}$ respecto al vector $b_3^{(3)}$:

$$\begin{aligned}
\nu_3 &:= \\
b_3^{(3)} &:= \\
\big[\lambda_{3,4}\big] &:= \\
\big[b_4^{(3)}\big] &:=
\end{aligned}$$

En el paso $p = 4$ calculamos $b_4^{(4)}$ como $b_4^{(3)}$ normalizado:

$$\begin{aligned}
\nu_4 &:= \\
b_4^{(4)} &:=
\end{aligned}$$

9. Programa para el caso de cuatro vectores. Dados cuatro vectores $a_1, a_2, a_3, a_4 \in \mathbb{R}^5$, construimos cuatro vectores ortonormales $b_1, b_2, b_3, b_4 \in \mathbb{R}^5$ aplicando el algoritmo de Gram–Schmidt modificado a los vectores dados a_1, a_2, a_3, a_4. Al final comprobamos que los vectores b_1, b_2, b_3, b_4 son ortogonales a pares y calculamos su matriz de Gram. El siguiente código se puede guardar en un archivo `MGSexample4.m`.

```
function [] = MGSexample4(),
   a1 = [-2; 5; 1; -1; 4]; a2 = [-1; 6; 3; 4; 3];
   a3 = [-11; 20; 9; 7; 2]; a4 = [4; 2; 1; -2; 5];
   # Copy:
   b1 = a1; b2 = a2; b3 = a3; b4 = a4;

   # Step 1
   nu1 = norm(b1);
   b1 = b1 / nu1;
   la12 = b1' * b2
   la13 =
   la14 =
   b2 = b2 - la12 * b1
   b3 =
   b4 =

   # Step 2
   nu2 = norm(b2)
   b2 =
   la23 = b2' * b3
   la24 =
   b3 =
   b4 =

   # Step 3
   nu3 =
   b3 =
   la34 =
   b4 =

   # Step 4
   nu4 =
   b4 =

   # Verify that the obtained vectores form an orthonormal list:
   B = [b1, b2, b3, b4]; G = B' * B
end
```

10. Código en forma matricial. Modificar el programa anterior de la siguiente manera: guardar los vectores en una matriz, guardar los coeficientes "lambdas" en un renglón.

```
function [] = MGSexample4matr(),
   a1 = [-2; 5; 1; -1; 4]; a2 = [-1; 6; 3; 4; 3];
   a3 = [-11; 20; 9; 7; 2]; a4 = [4; 2; 1; -2; 5];
   # Copy:
   B = [a1, a2, a3, a4];

   # Step 1
   nu1 = norm(B(:, 1));
   B(:, 1) = B(:, 1) / nu1;
   Lambdas1 = B(:, 1)' * B(:, 2 : 4);
   B(:, 2 : 4) = B(:, 2 : 4) - B(:, 2 : 4) * Lambdas1;

   ...
end
```

11. Programar el algoritmo modificado de Gram–Schmidt en forma matricial.

```
function [B] = MGS(A),
   [n, m] = size(A);
   B = A;
   for p = 1 : ???,
      nu = ???;
      B(:, ???) = ???;
      Lambdas = ???;
      B(:, ???) = ??? - ??? * Lambdas;
   end
end
```

Prueba:

```
function [] = testMGS(),
   m = 4; n = 5; A = rand(n, m);
   B = MGS(A); display(B' * B);
end
```

12. Complejidad del algoritmo modificado de Gram–Schmidt. Contar el número de multiplicaciones y divisiones que se realizan en la función `MGS`, si la matriz A es de tamaño $n \times m$. Se puede suponer que la función `norm`, aplicada a un vector de longitud n, hace n multiplicaciones.