PRÁCTICA SÉPTIMA: PROYECCIÓN ORTOGONAL. INVERSA DE MOORE-PENROSE

1. Proyección de un vector sobre un subespacio

En este apartado vamos a recordar como se proyecta un vector $\mathbf{v} \in \mathbb{R}^m$ sobre un subespacio vectorial L de $\mathbb{R}^m$.

1. Ejemplo : En primer lugar, generamos un conjunto de puntos en el espacio.

```
>> X=3*rand(3,100)-1;
```

Extraemos las coordenadas x, y y z.

```
>> x=X(1,:);
>> y=X(2,:);
>> z=X(3,:);
```

Dibujamos estos puntos en el espacio, y no cerramos la figura

```
>> plot3(x,y,z,'b.')
>> box on
>> grid on
>> hold on
```

Vamos a proyectar los puntos definidos por X sobre el subespacio vectorial de $\mathbb{R}^3$ generado por la columnas de

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & -1 \end{pmatrix}.$$

Introducimos en primer lugar la matriz A.

```
>> u1=[1;1;0];u2=[0;1;1];u3=[1;0;-1];
>> A=[u1,u2,u3];
```

Ahora calculamos la matriz de proyección. El comando `pinv` de `MATLAB` calcula la inversa de Moore-Penrose.

```
>> P=A*pinv(A)
```

Date: Mayo-2011.

Ahora, si multiplicamos la matriz X por la matriz P proyectaremos cada punto sobre el espacio de columnas de A.

```
>> PX=P*X;
```

Tomamos las componentes de cada punto.

```
>> Px=PX(1,:);
>> Py=PX(2,:);
>> Pz=PX(3,:);
```

Ya podemos dibujar los puntos originales y sus proyecciones.

```
>> plot3(Px,Py,Pz,'r.')
```

La pregunta es si realmente hemos conseguido lo que buscábamos. Es difícil de decir a partir de la figura obtenida. Sin embargo, podemos hacer dos cosas para convencernos de que la proyección se ha efectuado sobre el subespacio vectorial generado por los vectores $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$. Primero dibujemos los vectores $\mathbf{u}_1 = (1,1,0)^{\mathrm{t}}, \mathbf{u}_2 = (0,1,1)^{\mathrm{t}}$ y $\mathbf{u}_3 = (1,0,-1)$ sobre la figura con los siguientes comandos.

```
>> line([0,1],[0,1],[0,0],'linewidth',2,'color','k')
>> line([0,0],[0,1],[0,1],'linewidth',2,'color','k')
>> line([0,1],[0,0],[0,-1],'linewidth',2,'color','k')
>> hold off
```

El comando `line` permite añadir más gráficos sobre el dibujo. Los vectores $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$ aparecen en la nueva figura sobre el plano $\mathrm{im}(A)$.

Si ahora pulsamos el icono de rotación en la pantalla de la figura, podemos experimentar con diferentes puntos de vista. En la figura obtenida, usamos el ratón para colocar la figura con acimut 29 y elevación -40. Esto se puede hacer sin el ratón mediante el comando `view([29,-40])`. Vemos que los vectores $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$ se ocultan por la nube de puntos proyectados sobre el plano.

2. Soluciones aproximadas mínimo cuadráticas de sistemas de ecuaciones lineales

En algunas situaciones en las nos encontramos con sistema de ecuaciones $A\mathbf{x} = \mathbf{b}$, puede ser conveniente hallar un vector $\hat{\mathbf{x}}$ que este "cerca de ser solución del sistema"; entendiendo por esto que $A\hat{\mathbf{x}} - \mathbf{b}$ sea próximo a cero. Una de las formas más comunes de medir la proximidad de $A\hat{\mathbf{x}} - \mathbf{b}$ a cero es mediante el cálculo de la suma de los cuadrados de las componentes de $A\hat{\mathbf{x}} - \mathbf{b}$. Cualquier vector que minimice esta suma de cuadrados se llama **solución aproximada mínimo cuadrática**.

1. Ejemplo : Supongamos que queremos calcular la solución del sistema de ecuaciones lineales

$$\begin{aligned} m \cdot 0 + c &= 6 \\ m \cdot 1 + c &= 0 \\ m \cdot 2 + c &= 0 \end{aligned}$$

Este sistema está sobre-determinado: hay más ecuaciones que incógnitas. Es más, es incompatible.

```
>> M=[0,1,6;1,1,0;2,1,0]
>> R=rref(M)
```

La última fila de R representa la ecuación $0 \cdot m + 0 \cdot c = 1$, que no tiene solución.

Como es habitual el sistema se puede escribir en la forma

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} m \\ c \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix},$$

o bien $A\mathbf{x} = \mathbf{b}$, donde

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} m \\ c \end{pmatrix}, \mathbf{v} = \begin{pmatrix} 6 \\ 0 \\ 0 \end{pmatrix}.$$

Como el sistema no tiene solución, $\mathbf{b}$ no puede escribirse como combinación lineal de las columnas de A; en otras palabras, $\mathbf{b} \notin \mathrm{im}(A)$.

Teniendo en cuenta que una recta en el plano es de la forma $y = mx+c$, podemos reenunciar nuestro problema en términos geométricos como el de calcular una recta que se ajuste lo mejor posible, en sentido mínimo cuadrático, a los datos de la siguiente tabla:

x	0	1	2
y	6	0	0

Si dibujamos los datos de la tabla como puntos en un plano

```
>> plot([0,1,2],[6,0,0],'s')
>> axis([-1,7,-1,7])
>> grid on
>> hold on
```

se ve claramente que los puntos no están alineados, por lo que no es posible dibujar una recta a través de ellos como ya sabíamos. De modo que tendremos que contentarnos con hallar una solución aproximada.

Vamos a calcular la solución aproximada mínimo cuadrática de nuestro sistema. Para ello, si calculamos la proyección ortogonal $\mathbf{b}' = A\mathbf{b}''$ de $\mathbf{b}$ en $\mathrm{Im}\, A$, entonces la solución aproximada es $\mathbf{b}''$, que por definición de la pseudo inversa de Moore-Penrose, $\mathbf{b}'' = pinv(A)\mathbf{b}$.

```
>> A = [0,1;1,1;2,1]
>> b = [6;0;0]
>> xgorro  = pinv(A)*b
```

Nota.- Aunque sabemos que el sistema $A\mathbf{x} = \mathbf{b}$ es incompatible, observemos la salida de la siguiente sentencia.

```
>> A\b
```

Es la solución $\hat{\mathbf{x}}$ que habíamos obtenido. Esto ocurre porque el comando \ calcula la solución mínimo cuadrática del sistema $A\mathbf{x} = \mathbf{b}$. Teclea `help mldivide` para una descripción más completa.

En términos geométricos la solución aproximada mínimo cuadrática, $\hat{\mathbf{x}}$, obtenida nos da la ecuación de la recta que andábamos buscando. Como $m = -3$ y $b = 5$, la ecuación de la recta que mejor se ajusta es $y = -3x + 5$.

```
>> x=linspace(-1,2)
>> plot(x,-3*x+5,'r')
>> hold off
```

Es interesante examinar el error cometido al aproximar los datos con la recta de mejor ajuste. Los puntos originales eran $(0, 6), (1, 0)$ y $(2, 0)$, y sus proyecciones ortogonales sobre la recta son $(0, 5), (1, 2)$ y $(2, -1)$, respectivamente. Así, tenemos que n $x = 0$, el valor del dato es $y = 6$, y el punto sobre la recta correspondientes es $\hat{y} = 5$; entonces, el error cometido es $y - \hat{y} = 6 - 5 = 1$. Análogamente, en $x = 1$ tenemos que $y - \hat{y} = 0 - 2 = -2$, y en $x = 2$ obtenemos $y - \hat{y} = 0 - (-1) = 1$. Realmente estos errores se pueden calcular directamente con el vector $\mathbf{e} = \mathbf{b} - \mathbf{b}'$.

```
>> e=b-bb
```

Por tanto, el error total cometido es

```
>> norm(e)^2
```

2.1. **Otros ejemplos con MATLAB.**

2. Ejemplo : Supongamos que en un experimento físico, colgamos unas masas de un muelle, y medimos la distancia que el muelle elonga desde su punto de equilibrio para cada masa. Los datos los tenemos en la siguiente tabla.

m	10	20	30	40	50	60
d	1.4	2.8	3.6	5.0	6.4	7.2

Vamos a usar MATLAB para calcular la curva más simple que mejor ajusta a los datos de la tabla anterior.

En primer lugar, introducimos los datos en MATLAB y los dibujamos.

```
>> clear all
>> close all
>> m=(10:10:60)';
```

```
>> d=[1.4, 2.8, 3.6, 5.0, 6.4, 7.2]';
>> plot(m,d,'*')
>> hold on
```

Usamos el operador de transposición para formar vectores columna. Se ve en la figura que existe una tendencia lineal. En concreto,

```
>> corrcoef(m,d)
```

indica que el coeficiente de correlación es 0.9969.

Vamos a ajustar los datos con una recta de la forma $d = a + b\,m$. Primero, sustituimos cada punto en la ecuación:

$$\begin{aligned} 1.4 &= a + b \cdot 10 \\ 2.8 &= a + b \cdot 20 \\ 3.6 &= a + b \cdot 30 \\ 5.0 &= a + b \cdot 40 \\ 6.4 &= a + b \cdot 50 \\ 7.2 &= a + b \cdot 60 \end{aligned}$$

y escribimos el sistema matricialmente.

$$\begin{pmatrix} 1 & 10 \\ 1 & 20 \\ 1 & 30 \\ 1 & 40 \\ 1 & 50 \\ 1 & 60 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1.4 \\ 2.8 \\ 3.6 \\ 5.0 \\ 6.4 \\ 7.2 \end{pmatrix},$$

$$A\mathbf{x} = \mathbf{d}$$

El vector $\mathbf{d}$ ya lo tenemos definido en MATLAB. La segunda matriz de A contiene los datos de masa almacenados en el vector $\mathbf{m}$.

```
>> A=[ones(size(m)),m]
```

Luego ya sólo nos queda calcular la solución aproximada mínimo cuadrática del sistema $A\mathbf{x} = \mathbf{b}$ tal y como hicimos en el ejemplo anterior

```
>> xgorro= pinv(A)*d
```

Nota.- Notemos de nuevo que

```
>> A\d
```

nos da la solución correcta.

Entonces $a = 0.2800$ y $b = 0.1177$. Con estos valores vamos a dibujar la recta de mejor ajuste en nuestra figura.

```
>> ygorro=xgorro(1)+xgorro(2)*m;
>> plot(m,ygorro,'r')
>> hold off
```

3. Ejemplo: En otro experimento, un cohete de juguete es lanzado al aire. La altura del cohete a instantes determinados aparece en la tabla siguiente.

t	5	10	15	20	25	30
s	722	1073	1178	1117	781	102

Debemos examinar los datos y decidir un modelo apropiado para su ajuste por mínimos cuadrados.

Empecemos introduciendo los datos en vectores columna **t** y **s**.

```
>> clear all
>> close all
>> t=(5:5:30)';
>> s=[722, 1073, 1178, 1117, 781, 102]';
```

Podemos dibujar nuestros datos como sigue:

```
>> plot(t,s,'bs','MarkerFaceColor','b')
>> hold on
```

Aparentemente los datos forman una parábola. Intentemos entonces ajustar los datos a una ecuación de la forma $s = a + bt + ct^2$. Sustituimos los datos de la tabla en la ecuación $s = a + bt + ct^2$.

$$\begin{aligned} 722 &= a + b\cdot 5 + c\cdot (5)^2 \\ 1073 &= a + b\cdot 10 + c\cdot (10)^2 \\ 1178 &= a + b\cdot 15 + c\cdot (15)^2 \\ 1117 &= a + b\cdot 20 + c\cdot (20)^2 \\ 781 &= a + b\cdot 25 + c\cdot (25)^2 \\ 102 &= a + b\cdot 30 + c\cdot (30)^2 \end{aligned}$$

La expresión matricial del sistema es de la forma

$$\begin{pmatrix} 1 & 5 & 5^2 \\ 1 & 10 & 10^2 \\ 1 & 15 & 15^2 \\ 1 & 20 & 20^2 \\ 1 & 25 & 25^2 \\ 1 & 30 & 30^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 722 \\ 1073 \\ 1178 \\ 1117 \\ 781 \\ 102 \end{pmatrix},$$

$$A\mathbf{x} = \mathbf{s}.$$

Podemos introducir en MATLAB los valores de A de una forma sencilla.

```
>> A=[ones(size(t)),t,t.^2]
```

Vamos entonces a calcular la solución aproximada mínimo cuadrática del sistema $A\mathbf{x} = \mathbf{s}$.

```
>> xgorro = A\s
```

Entonces $a = 80.2000, b = 149.7814$ y $c = -4.9386$. Con estos coeficientes vamos a pintar la parábola de mejor ajuste. Además, queremos hacer dos estimaciones. Por un lado, vamos a averiguar la altura inicial del cohete, y por otro queremos saber en qué momento volvió a tierra. Por ello, extendemos el intervalo de t para que nos aparezcan esos datos.

```
>> tt=linspace(0,35);
>> sgorro=xgorro(1)+xgorro(2)*tt+xgorro(3)*tt.^2;
>> plot(tt,sgorro)
>> grid
>> hold off
```

El vector de errores es igual a $\mathbf{e} = \mathbf{s} - A\hat{\mathbf{x}}$, y podemos calcular su norma.

```
>> p=A*xgorro;
>> e=s-p
>> norm(e)
```

Finalmente, podemos preguntarnos por qué no realizamos, por ejemplo, un ajuste con una cúbica. La ecuación buscada es $s = a + bt + ct^2 + dt^3$ y, en ese caso, el sistema queda de la forma

$$\begin{pmatrix} 1 & 5 & 5^2 & 5^3 \\ 1 & 10 & 10^2 & 10^3 \\ 1 & 15 & 15^2 & 15^3 \\ 1 & 20 & 20^2 & 20^3 \\ 1 & 25 & 25^2 & 25^3 \\ 1 & 30 & 30^2 & 30^3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 722 \\ 1073 \\ 1178 \\ 1117 \\ 781 \\ 102 \end{pmatrix},$$

$$B\mathbf{x} = \mathbf{s}.$$

Veamos qué resulta siguiendo los pasos anteriores.

```
>> B=[ones(size(t)),t,t.^2,t.^3]
>> xgorroB=B\s
```

Observamos que el coeficiente d es de orden de 10^{-2}, lo que nos dice que la aportación de término en t^3 es pequeña. Si calculamos el error cometido, debe salir más pequeño que en el ajuste por una parábola.

```
>> eB=s-B*xgorroB;
>> norm(eB)
```

Por ello, no se trata de encontrar el modelo que dé el menor error, sino el que sea más sencillo y nos permita construir un modelo teórico.