Regresión de mínimos cuadrados

Por:
Ian Gabriel Cañas Fernández, 1092228

.

Profesor: Juan S. Pérez R.,
Asignatura: INL367L, Secc 01

Resumen:

En el presente laboratorio se planea computar los pesos de una regresión lineal usando el algoritmo Least Squares, a partir de el cual se hará una estimación de la salida del dataset obtenido. A partir del comportamiento del dataset que asumimos como lineal se desarrollará una regresión polinómica para observar la relación entre ambos métodos y las posibles variaciones de este último.

Ejercicios previos:

P6.1 Regresión lineal.

El objetivo principal de esta sección es el de computar los pesos de la regresión lineal a la simulación de un set de datos generado por el usuario. Este set sería segmentado en datos de entrenamiento y datos de prueba, siendo generado mediante la siguiente expresión:

$$r(t) = f(x) + e, \qquad donde\ f(x) = w_1 X + w_0$$

Se puede observar de antemano que la expresión simularía la medición ruidosa de un par de características con correlación lineal. Los valores seleccionados para la expresión fueron:

$$\boldsymbol{w} = [-0.7\ 220];$$
$$X \in \{1{:}\,1000\};$$
$$e \sim U(0{,}11).$$

P6.2 Predicción de las MPG.

Por otro lado, se espera la segmentación y predicción de un set de datos de Auto MPG Data Set. Se procurará computar los pesos de la regresión polinómica mediante el uso de Least Squares:

1. Considerando únicamente los dos *featucres* que arrojaron mejor desempeño.
2. Para 2 distintos sets de entrenamiento y pruebas.
3. Para 6 distintos grados de polinomios.

E6.1 Naive Bayes; predicción de la supervivencia.

En la primera sección se genera, aleatoriza y segmenta el set de datos como se conoce de antemano mediante lo presentado a continuación:

```
%%
X = 1:1000;
X = X';
w = [-0.7 220];
Y = sum ([ones(length(X), 1) X].*repmat(w, length(X), 1), 2);
Y = Y + normrnd(0, 11, length(X), 1);

data = [X Y];

dataset = data(randperm(length(X)), :); % "Dataset" porque son los datos con los que trabajaremos

i = round(length(X)*0.75);
D1 = ones(i, 1);

traindata = dataset(1:i, :);
testdata = dataset((i+1):end, :);
```

Más tarde, se genera una regresión lineal mediante la siguiente ecuación:

$$\boldsymbol{w} = (D^T D)^{-1}(D^T \boldsymbol{r})$$

Continuando con la aplicación de los pesos obtenidos al set de pruebas de los datos.

```
X_train = [ones(i,1) traindata(:,1)];
r = traindata(:, 2);

wA1 = inv(X_train'*X_train)*(X_train'*r)

X_test = [ones(length(testdata(:,1)),1) testdata(:,1)];
YML = X_test*wA1;

e1 = [YML testdata(:,2) (YML-testdata(:,2)).^2];
emean1 = mean(e1(:,3))
```

E6.2 Naive Bayes; predicción de la supervivencia.

En la segunda sección, igualmente, empezamos aleatorizando los datos y seccionándolos en sets de entrenamiento y pruebas.

```
%%
data2 = readtable('auto-mpg.csv');
dataset2 = data2(randperm(length(data2.mpg)), :);
% "Dataset" porque son los datos con los que trabajaremos

i = round(length(dataset2.mpg)*0.75);

traindata2 = dataset2(1:i, :);
testdata2 = dataset2((i+1):end, :);
```

Preparamos las matrices de características junto a su vector bias para la regresión. Para la primera prueba, mediante el uso de regresión a tres dimensiones, se hace selección de las dos mejores características. El método utilizado para esta sección fue la de recorrer las combinaciones posibles de las permutaciones mediante forward search.

```
X2 = [ones(length(testdata2.cylinders(:)),1) testdata2.cylinders testdata2.displacement testdata2.weight testdata2.acceleration testdata2.model_year testdata2.origin];
X = [ones(i,1)  traindata2.cylinders traindata2.displacement traindata2.weight traindata2.acceleration traindata2.model_year traindata2.origin];
r = traindata2.mpg;

sel = zeros(length(X(1,:)), 1);
l=2;
b=2;
a=0;
c=0;
while a~=b
    c = c + 1;
    a=b;
    sel = zeros(length(X(1,:)), 1);
    for j = 2:length(X(1,:))
        if a==j
            continue
        else
            wA2 = inv(X(:, [1 a j])'*X(:, [1 a j]))*X(:, [1 a j])'*r;

            YML = X2(:, [1 a j])*wA2;

            e2 = [YML testdata2.mpg (YML-testdata2.mpg).^2 1-abs(YML-testdata2.mpg)./length(testdata2.mpg)];
            emean = mean(e2(:,3));
            eprop = mean(e2(:,4));
        end
        sel(j) = eprop;
    end
    l = find(sel==max(sel));
    l = min(l);
    b=l;
    if c==length(X(1,:))
        break
    end
end
```

Aplicamos regresión lineal y revisa error mínimo cuadrado:

```
wA2 = inv(X(:, [1 a b])'*X(:, [1 a b]))*X(:, [1 a b])'*r

YML = X2(:, [1 a b])*wA2;

e2 = [YML testdata2.mpg (YML-testdata2.mpg).^2 1-abs(YML-testdata2.mpg)./length(testdata2.mpg)];
emean2a = mean(e2(:,3))
eprop = mean(e2(:,4));
```

Finalmente, reprocesando los datos, se preparan las matrices de características y bias. Se implementa regresión de mínimos cuadrados para varios grados polinómicos entre hasta el grado seis (6):

```
%%

traindata2 = dataset2(1:i, :);
testdata2 = dataset2((i+1):end, :);

i = length(traindata2.cylinders);
j = length(testdata2.cylinders);

X2 = [ones(j,1) testdata2.cylinders testdata2.displacement testdata2.weight testdata2.acceleration t
X = [ones(i,1)  traindata2.cylinders traindata2.displacement traindata2.weight traindata2.accelerati
r = traindata2.mpg;

for d = 2:length(X(1,:))
    disp('Pesos y error para polinomio de grado ')
    disp(d-1)

    wA3 = inv(X(:,1:d)'*X(:,1:d))*X(:,1:d)'*r
    mpgML = X2(:,1:d)*wA3;

    e3 = [mpgML testdata2.mpg (mpgML-testdata2.mpg).^2];
    emean3 = mean(e3(:,3))
end
```

R6 Resultados:

Para la primera sección se obtuvo los siguientes resultados de pesos y error cuadrado promedio para varias corridas del código:

wA1 =	wA1 =	wA1 =	wA1 =
0.5444	-0.6428	-0.6717	0.2104
219.9985	220.0002	219.9995	219.9989
emean1 =	emean1 =	emean1 =	emean1 =
116.8407	94.5199	123.2305	128.3524

Para la segunda sección se trabaja con la mejor combinación de características para conocer su eficiencia:

```
wA2 =

  -14.9692
   -0.0069
    0.7773


emean2a =

   11.1919
```

Finalmente, se aplican las regresiones polinómicas:

```
Pesos y error para polinomio de grado
     1


wA3 =

   43.4585
   -3.6466


emean3 =

   20.7290
```

```
Pesos y error para polinomio de grado
     2


wA3 =

   36.0394
   -0.1168
   -0.0613


emean3 =

   20.9305
```

```
Pesos y error para polinomio de grado
     3


wA3 =

   43.6048
    0.2233
   -0.0247
   -0.0055


emean3 =

   17.8851
```

```
Pesos y error para polinomio de grado
     4


wA3 =

   39.6404
    0.1762
   -0.0154
   -0.0062
    0.2699


emean3 =

   18.5344
```

```
Pesos y error para polinomio de grado
     5


wA3 =

  -16.8786
    0.1316
    0.0026
   -0.0072
    0.1650
    0.7651


emean3 =

   11.7478
```

```
Pesos y error para polinomio de grado
     6


wA3 =

  -22.6243
   -0.0948
    0.0141
   -0.0072
    0.2400
    0.7801
    1.5054


emean3 =

   11.2473
```

A.5 Análisis:

En base a los resultados obtenidos mediante la regresión, se observó que los resultados de pesos obtenidos mediante la primera sección se veían directamente relacionados al error cuadrado promedio. Este error tiende más a parecer menor según los pesos de la predicción se asemejen a los computados anteriormente en la generación de los datos. Por lo que este método tiende a no ser muy alterado ante el ruido siempre y cuando este tenga su media en el cero.

Luego, comprobando lo obtenido con las dos mejores características y con varios grados de polinomios, se observó que los grados de polinomios afinan su eficiencia según se aumente dicho grado. Sin embargo, si se desea trabajar con pocas características representativas, es conveniente el uso de la mejor combinación, tomando en cuenta que trabajar con muchas características para este método no es computacionalmente tan pesado.