UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROIECT IDENTIFICAREA SISTEMELOR

Modelarea unei functii necunoscute

Grupa 30135 semigrupa 1

Student:

- Bârgăoanu Bogdan Alexandru

Grupa nr. 17 – set de date nr. 8

Profesor îndrumător: Lucian Buşoniu

1. Introducere

- **Scop** - Identificarea (găsirea modelului) unei funcții necunoscute prin intermediul căreia sunt generate datele de ieșire.
- **Date cunoscute** – Se dau un set de date de intrare , și un set de date de iesire.
- **Cerința** - Pornind de la condiția enunțată mai sus, se cere găsirea unei funcții de aproximare a funcției noastre necunoscute. Această funcție este de grad configurabil, așadar calculăm funcția pentru diferite valori ale gradului m, până când ajungem la un grad ideal, cel în urma caruia rezulta eroarea patratica medie cea mai mica.
- **Metoda utilizata** – regresie liniară.

2. Punerea problemei

Pornind de la formula $Y = \varphi * \theta$, luată în cazul cel mai simplu, unde Y reprezintă un vector coloană al ieșirilor $Y \in R^N$, φ este matricea de regresori

$\varphi \in R^{NxN}$, iar θ reprezintă vectorul coloană al coeficienților $\theta \in R^N$, va trebui să adaptăm Y , φ respectiv θ, pentru cazul nostru, unde Y reprezintă o suprafață dependentă de intrarea unui X1(i) si X2(j) , Y = f(X1(i),X2(j)).

Acest lucru ne va conduce să alegem Y ca fiind o matrice $Y \in R^{NxN}$, φ ca fiind o matrice 3D, $\varphi \in R^{NxNxM}$ formată din polinomul rezultant din toate combinațiile de puteri ale lui X1 și X2 până la puterea m, unde M reprezintă gradul de aproximare ales corespunzator lui m, și θ ca fiind o matrice $Y \in R^{NxN}$.

Pentru a ne ușura implementarea și calculele, vom redimensiona aceste matrici, fără ca proprietatea formulei inițiale să fie modificată. Așadar vom alege:

- $Y \in R^{(NxN)}$
- $\varphi \in R^{(NxN)xM}$
- $\Theta \in R^{(NxN)}$

unde N reprezintă lungimea lui X1 si X2.

3. Rezolvarea problemei

3.1. Funcția polinom

Este o funcție care calculează o linie din matricea de regresori.

Din cauza dimensiunilor alese, pe liniile matricei φ vom avea regresorii în forma următoare: //nota: indexare de la 1 pentru funcționalitate MATLAB

φ (1) = polinoamele generate de X1(1) si X2(1) ;

φ (2) = polinoamele generate de X1(1) si X2(2) ;

…

φ (N) = polinoamele generate de X1(1) si X2(N) ;

φ (N+1) = polinoamele generate de X1(2) si X2(1) ;

…

φ (2*N) = polinoamele generate de X1(2) si X2(1) ;

φ (2*N+1) = polinoamele generate de X1(2) si X2(2) ;

…

φ (N*N) = polinoamele generate de X1(N) si X2(N) ;

Observăm că indexul lui X1 corespunde multiplului lui N din φ , iar cel al lui X2 corespunde cu restul împărțirii la N al liniei din φ.

⇨ Pentru accesarea unui câmp aleator φ (i,j) care conține polinomul corespunzator lui X1(i) si X2(j) vom avea următoarea formulare:

φ (L) = (i-1)*lungime(X1) + j ;

unde i reprezinta coeficientul lui X1 si j cel al lui X2.

Codul integral pentru functia de generare polinom poate fi găsit mai jos:

```
function phi = genereazaPolinom(x1, x2, m)
    phi = [];

    for grad_x1 = 0:m
        for grad_x2 = 0:(m - grad_x1)
            termen = x1.^grad_x1.* x2.^grad_x2;
            phi = [phi, termen];
        end
    end
end
```

3.2. Identificarea modelului

Inițial am peluat datele de ientificare din fișierul dat, urmand ca apoi să afișăm graficul funcției datelor de ieșire.

Pentru găsirea matricei φ se parcurg toți indecșii lui X1 respectiv X2, folosindu-se funcția definită anterior.

```
for i = 1:length(x1_id)
        for j = 1:length(x2_id)
            polinom = genereazaPolinom(x1(i), x2(j), m);
             phi = [phi; polinom];
        end
end
```

Dupa ce matricea φ este generată corect, folosind formula $Y=\varphi\Theta$, vom determina vectorul de coeficienți.

$$\Theta = \varphi \setminus Y ;$$

3.3. Validarea modelului

Pentru validare vom avea nevoie și de matricea φ_{val} generată din datele de validare. Se procedează exact la fel ca și anterior, doar că se vor introduce ca input la funcție datele corecte.

Urmează să se genereze apoi ieșirea aproximată, folosind din nou formula fundamentală, dar acum pentru a afla Y_{aprox}. Acest lucru se va realiza și pentru identificare și pentru validare.

$$\mathbf{Y_{aprox} = \varphi_{val} * \Theta ;}$$

Având aproximarea, acum se poate calcula eroare pentru fiecare Y, respectiv MSE (eroarea medie pătratică) pentru fiecare grad de aproximare.

$$\mathbf{e = Y - Y_{aprox} ;}$$

$$\mathbf{MSE = \frac{1}{N*N} * \sum e^2 ;}$$

Considerații:

Codul este configurat să itereze prin toate gradele de aproximare și să realizeze toți pașii anteriori de la gradul de aproximare 1 până la M. Deci la fiecare pas, eroare se va recalcula pentru noile valori , si MSE se va stoca în doi vectori de lungime M, stocând eroarea medie patratica pentru fiecare grad de aproximare. De asemenea, se va stoca si Y_{aprox} cu MSE minim, acel semnal fiind considerat optim. Ca un ultim pas, urmează afișarea graficelor cu valorile optime.

4. Grafice si MSE

4.1. Date identificare si validare

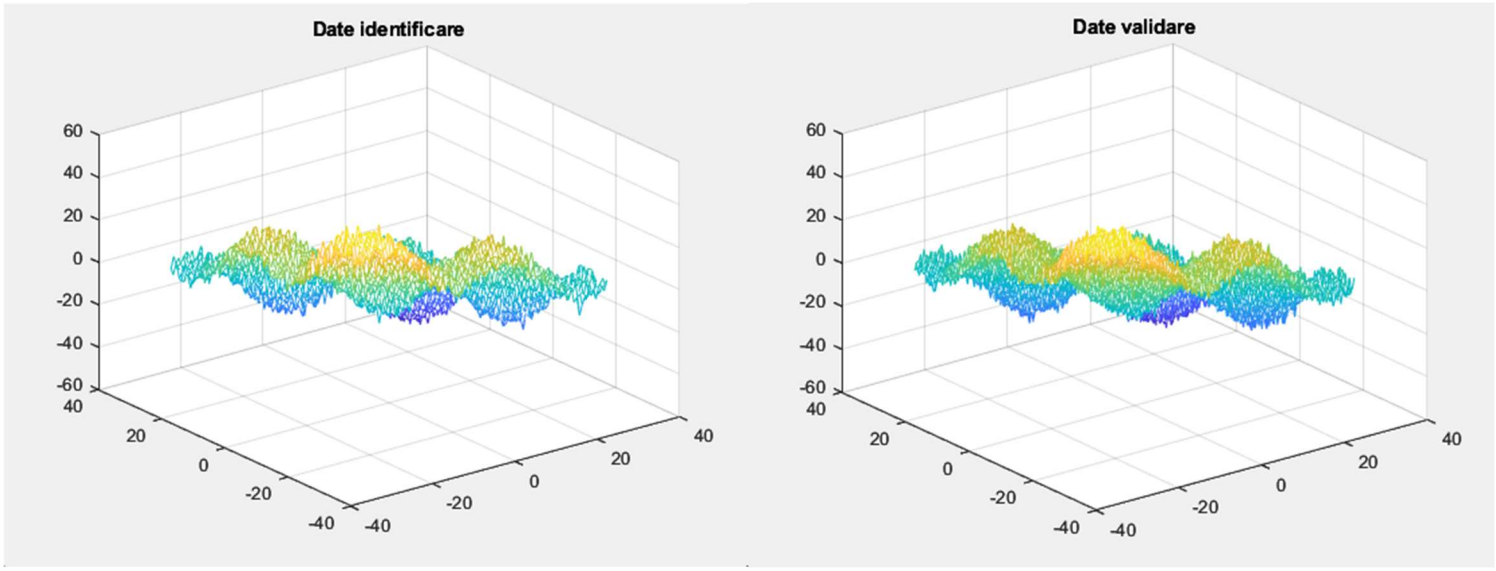

Figura 1: Date identificare si validare

4.3. Y_{aprox} optim

Y_{aprox} de validare a fost obținut la un grad de aproximare M = 7, pentru un loop de training pana la un grad maxim 15. Respectiva aproximare a ieșirii are o MSE = 6.624662885800823 iar Y_{aprox} de identificare a fost obținut la M = 8 cu MSE =6.521527555825020.

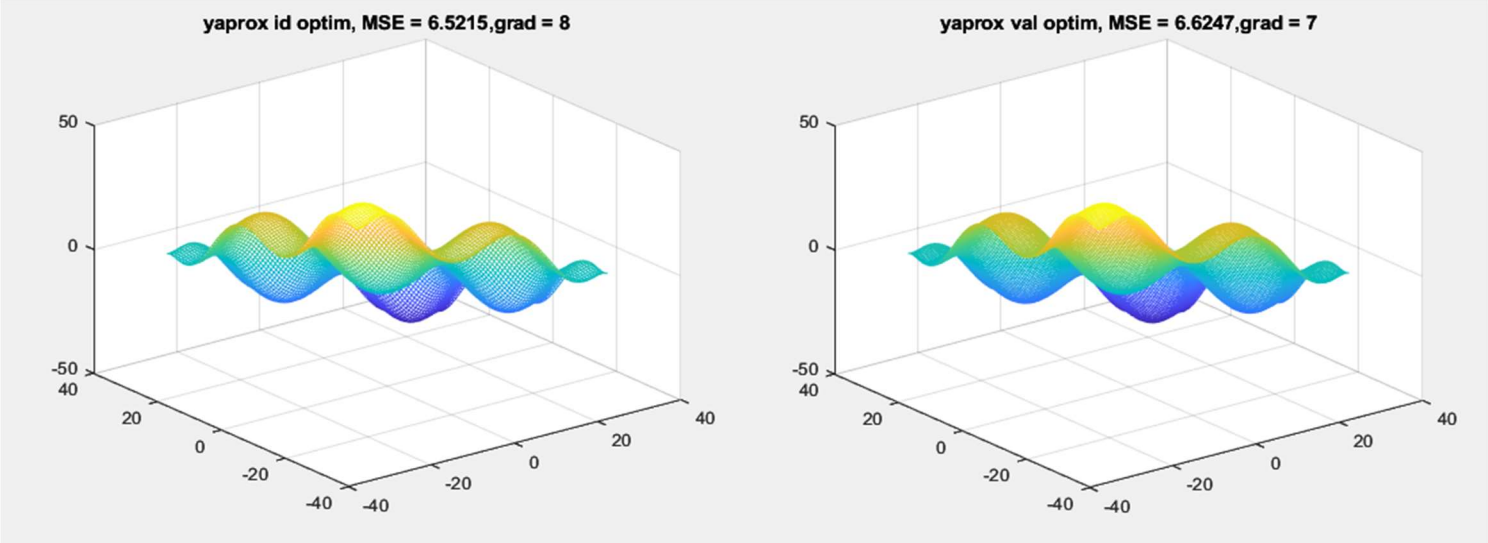

Figura 2: Y aprox pentru identificare si validare

4.4. MSE pentru identificare și validare

Din graficul următor care reprezintă MSE-ul pentru datele de identificare (rosu), respectiv pentru cele de validare(albastru) se poate observa cum inițial eroare ieșirii scade cu cresterea gradului de aproximare, dar in jurul gradului 8 (~sau 7) incepe fenomenul de overfitting, care consta in modelarea zgomotelor din setul de date.

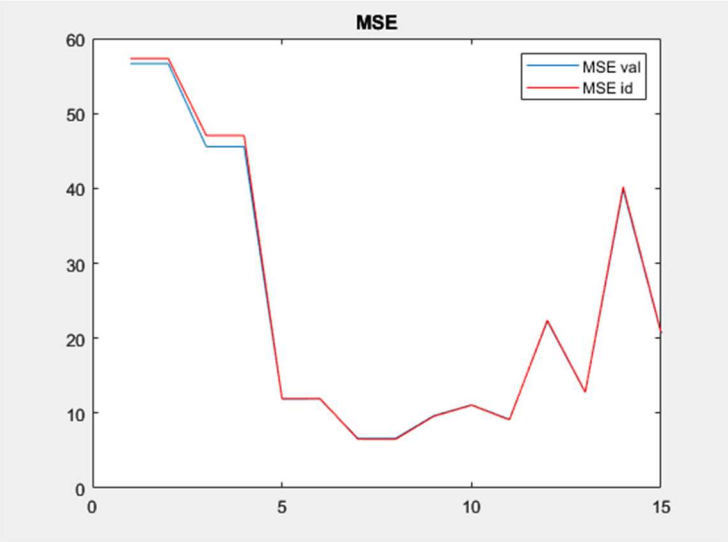

Figura 3: MSE

5. Cod complet

```
load('proj_fit_08');
M = 15;
MIN_MSE_val = inf;
MIN_MSE_id = inf;
%date identificare
x1_id = id.X{1};
x2_id = id.X{2};
y_id = id.Y;

%date validare
x1_val = val.X{1};
x2_val = val.X{2};
y_val = val.Y;

figure;
mesh(x1_id,x2_id,y_id);
title('Date identificare');
figure;
mesh(x1_val,x2_val,y_val);
title('Date validare');

%initializare MSE si y_final ca fiind goale inafara loop-ului de training pentru
a nu pierde valorile
MSE_val = [];
MSE_id = [];
y_final_val = [];
y_final_id = [];
for z = 1:M

%initializare matrici necesare ca fiind goale
phi_id = [];
phi_val = [];
theta = [];

%phi identificare
for i = 1:length(x1_id)
    for j = 1:length(x2_id)
        polinom = genereazaPolinom(x1_id(i), x2_id(j), z);
        phi_id = [phi_id; polinom];
    end
end
%phi validare
for i = 1:length(x1_val)
    for j = 1:length(x2_val)
        polinom = genereazaPolinom(x1_val(i), x2_val(j), z);
        phi_val = [phi_val; polinom];
    end
end
%phi_3d = reshape(phi,[length(x1_id),length(x2_id),width(phi)]);
y_concatenat = reshape(y_id,[],1);
%generare iesirii aproximate
```

```
theta = phi_id\y_concatenat;
y_aprox_id = phi_id * theta;
y_aprox_val = phi_val * theta;
y_aprox2d_val = reshape(y_aprox_val,[length(y_val),width(y_val)]);
y_aprox2d_id = reshape(y_aprox_id,[length(y_id),width(y_id)]);

%calcul MSE
e_val = y_val-y_aprox2d_val;
e_val = reshape(e_val,[length(e_val)*length(e_val),1]);
MSE_val(z) = 1/length(e_val) * sum(e_val.^2);
e_id = y_id - y_aprox2d_id;
e_id = reshape(e_id,[length(e_id)*length(e_id),1]);
MSE_id(z) = 1/length(e_id) * sum(e_id.^2);

if MSE_val(z) < MIN_MSE_val
y_final_val = y_aprox2d_val;
MIN_MSE_val = MSE_val(z);
grad_minim_val = z;
end

if MSE_id(z) < MIN_MSE_id
y_final_id = y_aprox2d_id;
MIN_MSE_id = MSE_id(z);
grad_minim_id = z;
end
end

%afisare rezultate finale
figure;
mesh(x1_val,x2_val,y_final_val);
title(['yaprox val optim, MSE = ',num2str(MIN_MSE_val),',grad =
',num2str(grad_minim_val)]);


figure;
mesh(x1_id,x2_id,y_final_id);
title(['yaprox id optim, MSE = ',num2str(MIN_MSE_id),',grad =
',num2str(grad_minim_id)]);

figure;
plot(MSE_val);
hold on
plot(MSE_id,'Color','Red')
legend('MSE val','MSE id')
title('MSE');

function phi = genereazaPolinom(x1, x2, m)
    phi = [];

    for grad_x1 = 0:m
        for grad_x2 = 0:(m - grad_x1)
            termen = x1.^grad_x1.* x2.^grad_x2;
            phi = [phi, termen];
        end
    end
```

```
end
```

6. Concluzii

În concluzie, folosirea aproximatorului polinomial este o metodă eficientă pentru a obține o aproximare a unei ieșiri. Totuși, cel mai important lucru în această metodă este selecția gradului de aproximare pentru a obține eroarea minimă pentru model. De asemenea este de luat în calcul că un grad prea mare poate duce la overfitting, cum s-a putut observa în Figura 3. , deci abordarea cea bună este găsirea unui grad de aproximare care să nu fie prea mic și să conducă la erori mari, dar nici prea mare pentru a nu cauza problema de overfitting. Acest fapt a condus la conceperea codului sursă, în ideea de a itera prin cât mai multe grade posibile, având ca scop găsirea gradului cu eroare minima, fără a fi nevoie de un user input la fiecare schimbare de grad.