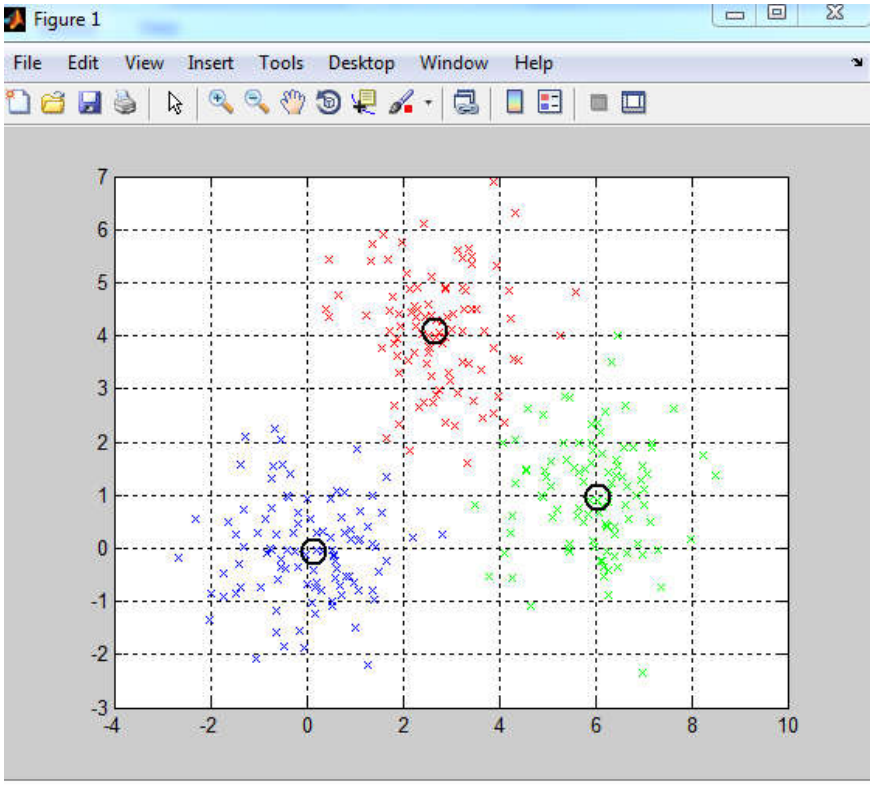
حل مسئله خوشه بندی با الگوریتم زنبورعسل bee در متلب

صورت مسئله:

یک دیتاست مشخص داریم در فایلmydata.m که شامل ۳۰۰ عضو ۲ بعدی است می خواهیم ۳ تا نقطه ۲ بعدی به عنوان مرکز خوشه پیدا کنیم به طوری که فاصله هر عضو تا مرکز خوشه آن مینیمم باشد.



شرح کد:

این سورس کد شامل ۴ فایل می باشد که عبارتند از:

Clusteringfit.m: این تابع برای محاسبه برازندگی می باشد به این صورت که کمترین فاصله هر عضو از مراکز داده را در نظر می گیرد و آن ها را با هم جمع می کند چون این مقدار باید کم شود ولی ما می خواهیم برازندگی را حساب کنیم از عکس این مقدار به اضافه یک در مخرج استفاده می کنیم.

```
function [z, out] = Clusteringfit(m, X)

    % محاسبه ماتریس فاصله
    d = pdist2(X, m);

 % خوشه ها را اختصاص دهید و نزدیکترین فاصله ها را پیدا کنید
    [dmin, ind] = min(d, [], 2);

    % مجموع فاصله درون خوشه ای
    WCD = sum(dmin);

    z=1/(1+WCD);      تبدیل هزینه به برازندگی

                                  دادن مقادیر حساب شده به خروجی
    out.d=d;
    out.dmin=dmin;
    out.ind=ind;
    out.WCD=WCD;

end
```

RouletteWheelSelection.m: احتمال انتخاب شدن هر زنبور برای پیشاهنگی

```
function i = RouletteWheelSelection(P)

    r = rand;

    C = cumsum(P);

    i = find(r <= C, 1, 'first');

end
```

PlotSolution.m: کشیدن بهترین راه حل

```
function PlotSolution(X, sol)

    % مراکز خوشه
    m = sol.Position;
    k = size(m,1);
```

```
    % شاخص های خوشه ای
    ind = sol.Out.ind;

    Colors = hsv(k);

    for j=1:k
        Xj = X(ind==j,:);

plot(Xj(:,1),Xj(:,2),'x','LineWidth',1,'Color',Colors(j,:))
;
        hold on;
    end

    plot(m(:,1),m(:,2),'ok','LineWidth',2,'MarkerSize',12);

    hold off;
    grid on;

end
```

abc.m: حل مسیله خوشه بندی با پیاده سازی الگوریتم زنبور عسل مصنوعی

```
clc;
clear;
close all;

%% تعریف مسیله
data = load('mydata'); فایل داده ها
X = data.X;
k = 3; تعداد مراکز خوشه

fitFunction=@(m) Clusteringfit(m, X);     % تابع برازندگی

VarSize=[k size(X,2)];  % اندازه ماتریس متغیرهای تصمیم

nVar=prod(VarSize);     % تعداد متغیرهای تصمیم

VarMin= repmat(min(X),k,1);       % حد بالا متغیرها
VarMax= repmat(max(X),k,1);       % حد پایین متغیرها


%% تنظیمات پارامتر الگوریتم زنبور عسل

MaxIt = 100;              % ماکزیمم تعداد تکرار
```

```
nPop = 100;                 % اندازه جمعیت(اندازه کلونی
(
nOnlooker = nPop;           % تعداد زنبورهای پیشاهنگ

L = round(0.6*nVar*nPop); %  پارامتر محدودیت ترک (حد
آزمایشی)

a = 1;                      % ضریب شتاب بالا محدود

%% مقداردهی اولیه

% ساختار زنبور عسل خالی
empty_bee.Position = [];
empty_bee.fit = [];
empty_bee.Out = [];
% آرایه جمعیت را مقداردهی اولیه کنید
pop = repmat(empty_bee, nPop, 1);

% بهترین راه حل تاکنون پیدا شده را مقداردهی اولیه کنید
BestSol.fit = -inf;

% ایجاد جمعیت اولیه
for i = 1:nPop
    pop(i).Position = unifrnd(VarMin, VarMax, VarSize);
    [pop(i).fit,  pop(i).Out]=
fitFunction(pop(i).Position);
    if pop(i).fit >= BestSol.fit
        BestSol = pop(i);
    end
end

% شمارنده انصراف
C = zeros(nPop, 1);

% آرایه برای نگه داشتن بهترین مقادیر مناسب
Bestfit = zeros(MaxIt, 1);

%% حلقه اصلی الگوریتم زنبور عسل

for it = 1:MaxIt

    % زنبورهای کارگر
    for i = 1:nPop
```

```
            % k نباشد i را به طور تصادفی انتخاب کنید ، برابر با
            K = [1:i-1 i+1:nPop];
            k = K(randi([1 numel(K)]));

            % ضریب شتاب را تعریف کنید
            phi = a*unifrnd(-1, +1, VarSize);

            % مکان زنبور جدید
            newbee.Position =
pop(i).Position+phi.*(pop(i).Position-pop(k).Position);

            % افزودن مرزها
            newbee.Position = max(newbee.Position, VarMin);
            newbee.Position = min(newbee.Position, VarMax);

            % ارزیابی
            [newbee.fit, newbee.Out]=
fitFunction(newbee.Position);

            % مقایسه
            if newbee.fit >= pop(i).fit
                pop(i) = newbee;
            else
                C(i) = C(i)+1;
            end

        end

        % مقادیر تناسب اندام و احتمال انتخاب را محاسبه کنید
        F = zeros(nPop, 1);
        Meanfit = mean([pop.fit]);
        for i = 1:nPop
            F(i) = exp(-pop(i).fit/Meanfit); % احتمال برازندگی
        end
        P = F/sum(F);

        % زنبورهای پیشاهنگ
        for m = 1:nOnlooker

            % مکان منبع را انتخاب کنید
            i = RouletteWheelSelection(P);

             k را به طور تصادفی انتخاب کنید ، برابر با i نباشد
            K = [1:i-1 i+1:nPop];
```

```
            k = K(randi([1 numel(K)]));

            % ضریب شتاب را تعریف کنید
            phi = a*unifrnd(-1, +1, VarSize);

            % مکان زنبور جدید
            newbee.Position =
pop(i).Position+phi.*(pop(i).Position-pop(k).Position);

            % افزودن مرزها
            newbee.Position = max(newbee.Position, VarMin);
            newbee.Position = min(newbee.Position, VarMax);

            % ارزیابی
            [newbee.fit, newbee.Out]=
fitFunction(newbee.Position);

            % مقایسه
            if newbee.fit >= pop(i).fit
                pop(i) = newbee;
            else
                C(i) = C(i) + 1;
            end

        end

        % زنبورهای پیشاهنگی
        for i = 1:nPop
            if C(i) >= L
                pop(i).Position = unifrnd(VarMin, VarMax,
VarSize);
                [pop(i).fit,pop(i).Out] =
fitFunction(pop(i).Position);
                C(i) = 0;
            end
        end

        % به روز رسانی بهترین راه حل که تاکنون پیدا شده است
        for i = 1:nPop
            if pop(i).fit >= BestSol.fit
                BestSol = pop(i);
            end
        end
```

```
    % ذخیره بهترین مناسب تا به حال پیدا شده است
    Bestfit(it) = BestSol.fit;

    % نمایش اطلاعات تکرار
    disp(['Iteration ' num2str(it) ': Best fit = '
num2str(Bestfit(it))]);

    % طرح راه حل
    figure(1);
    PlotSolution(X, BestSol);
    pause(0.01);
end

%% نتایج

figure;
semilogy(Bestfit, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best fit');
grid on;
```

در ادامه نتایج...

نتایج:

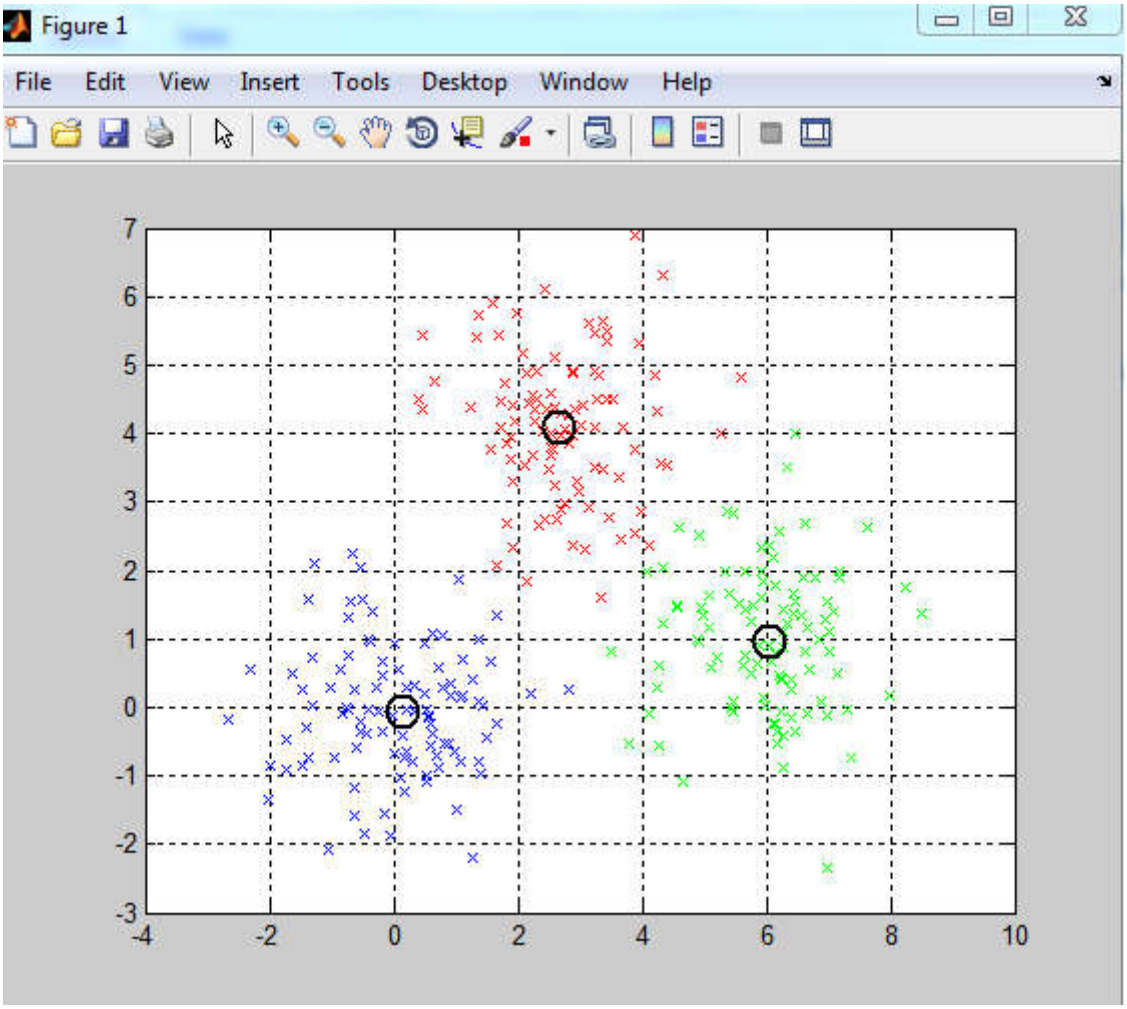
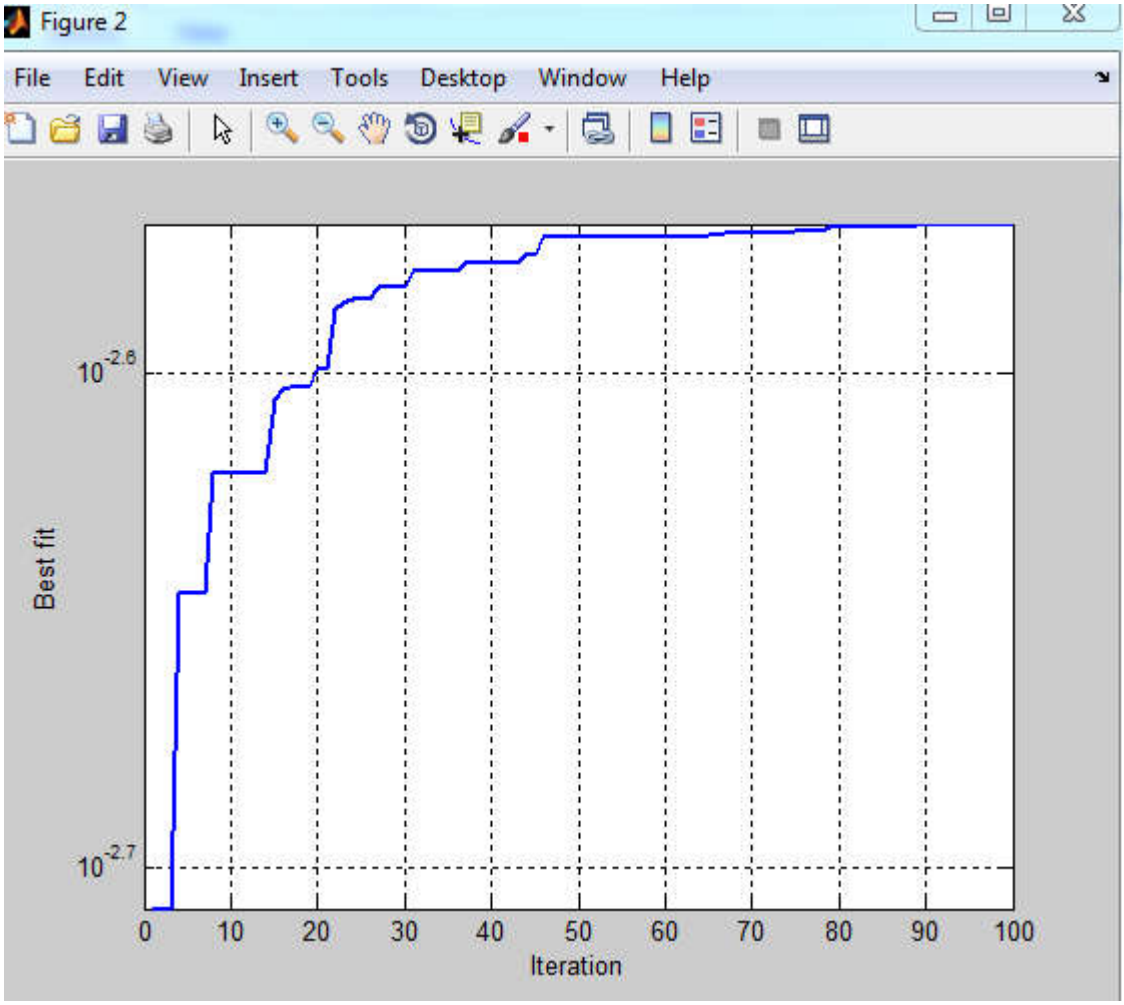


Figure 1
File Edit View Insert Tools Desktop Window Help
7
6
5
4
3
2
1
0
-1
-2
-3
-4 -2 0 2 4 6 8 10




```
Iteration 1: Best fit = 0.0019577

Iteration 2: Best fit = 0.0019577

Iteration 3: Best fit = 0.0019577

Iteration 4: Best fit = 0.002267

Iteration 5: Best fit = 0.002267

Iteration 6: Best fit = 0.002267

Iteration 7: Best fit = 0.002267

Iteration 8: Best fit = 0.0023984
```

```
Iteration 9: Best fit = 0.0023984
Iteration 10: Best fit = 0.0023984
Iteration 11: Best fit = 0.0023984
Iteration 12: Best fit = 0.0023984
Iteration 13: Best fit = 0.0023984
Iteration 14: Best fit = 0.0023984
Iteration 15: Best fit = 0.0024822
Iteration 16: Best fit = 0.0024936
Iteration 17: Best fit = 0.0024972
Iteration 18: Best fit = 0.0024972
Iteration 19: Best fit = 0.0024972
Iteration 20: Best fit = 0.0025168
Iteration 21: Best fit = 0.0025168
Iteration 22: Best fit = 0.0025875
Iteration 23: Best fit = 0.0025953
Iteration 24: Best fit = 0.0026014
Iteration 25: Best fit = 0.0026014
Iteration 26: Best fit = 0.0026014
Iteration 27: Best fit = 0.0026146
Iteration 28: Best fit = 0.0026146
Iteration 29: Best fit = 0.0026146
Iteration 30: Best fit = 0.0026146
Iteration 31: Best fit = 0.002634
```

```
Iteration 32: Best fit = 0.002634
Iteration 33: Best fit = 0.002634
Iteration 34: Best fit = 0.002634
Iteration 35: Best fit = 0.002634
Iteration 36: Best fit = 0.002634
Iteration 37: Best fit = 0.0026442
Iteration 38: Best fit = 0.0026442
Iteration 39: Best fit = 0.0026442
Iteration 40: Best fit = 0.0026442
Iteration 41: Best fit = 0.0026442
Iteration 42: Best fit = 0.0026442
Iteration 43: Best fit = 0.0026442
Iteration 44: Best fit = 0.0026555
Iteration 45: Best fit = 0.0026555
Iteration 46: Best fit = 0.002676
Iteration 47: Best fit = 0.002676
Iteration 48: Best fit = 0.002676
Iteration 49: Best fit = 0.002676
Iteration 50: Best fit = 0.002676
Iteration 51: Best fit = 0.002676
Iteration 52: Best fit = 0.002676
Iteration 53: Best fit = 0.002676
Iteration 54: Best fit = 0.002676
```

```
Iteration 55: Best fit = 0.002676
Iteration 56: Best fit = 0.002676
Iteration 57: Best fit = 0.002676
Iteration 58: Best fit = 0.0026771
Iteration 59: Best fit = 0.0026771
Iteration 60: Best fit = 0.0026771
Iteration 61: Best fit = 0.0026771
Iteration 62: Best fit = 0.0026771
Iteration 63: Best fit = 0.0026771
Iteration 64: Best fit = 0.0026771
Iteration 65: Best fit = 0.0026787
Iteration 66: Best fit = 0.0026787
Iteration 67: Best fit = 0.0026833
Iteration 68: Best fit = 0.0026833
Iteration 69: Best fit = 0.0026833
Iteration 70: Best fit = 0.0026833
Iteration 71: Best fit = 0.0026833
Iteration 72: Best fit = 0.0026833
Iteration 73: Best fit = 0.0026833
Iteration 74: Best fit = 0.0026833
Iteration 75: Best fit = 0.0026856
Iteration 76: Best fit = 0.0026856
Iteration 77: Best fit = 0.0026856
```

```
Iteration 78: Best fit = 0.0026857
Iteration 79: Best fit = 0.0026888
Iteration 80: Best fit = 0.0026888
Iteration 81: Best fit = 0.0026888
Iteration 82: Best fit = 0.0026888
Iteration 83: Best fit = 0.0026888
Iteration 84: Best fit = 0.0026888
Iteration 85: Best fit = 0.0026888
Iteration 86: Best fit = 0.0026888
Iteration 87: Best fit = 0.0026888
Iteration 88: Best fit = 0.0026888
Iteration 89: Best fit = 0.0026921
Iteration 90: Best fit = 0.0026921
Iteration 91: Best fit = 0.0026921
Iteration 92: Best fit = 0.0026921
Iteration 93: Best fit = 0.0026921
Iteration 94: Best fit = 0.0026921
Iteration 95: Best fit = 0.0026921
Iteration 96: Best fit = 0.0026921
Iteration 97: Best fit = 0.0026921
Iteration 98: Best fit = 0.0026921
Iteration 99: Best fit = 0.0026921
Iteration 100: Best fit = 0.0026933
```