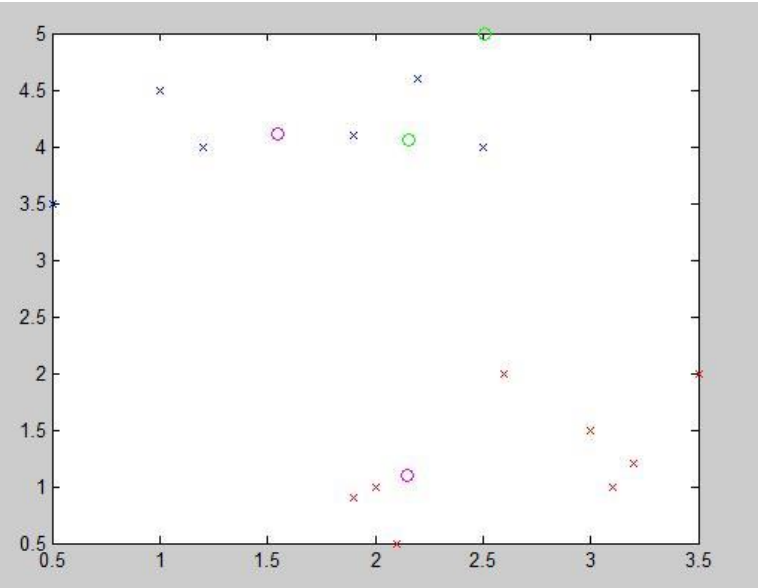
K-Means

This algorithm is viewed together with RBF (radial basis function).
By itself, it is an unsupervised learning algorithm because we do not need the target values.

Algorithm



We may have two different classes (each color is a class) which are clearly distinguishable, so we need two different centers.

At first, we randomly pick up two centers within the area, and we create two temporal clusters whose elements are the closest ones to each random center.

After that, we calculate the mean of each cluster and these means will represent our new centers. We again calculate the closest elements to these new centers and repeat the process until these means remain unchanged.

Code in Appendix 1.

Appendix 1: K means

```
clear;clc;clf;

intervalA = 0;
intervalB = 5;
samples = 10;

x1=[0.5 1 1.2 1.9 2.2 2.5]; % 6 samples
y1=[3.5 4.5 4 4.1 4.6 4];

x2 = [1.9 2 2.1 2.6 3 3.1 3.2 3.5]; % 8 samples
y2 = [0.9 1 0.5 2 1.5 1 1.2 2];

%x1 = intervalA + (intervalB-intervalA).*rand(1,samples/2);
%y1 = intervalA + (intervalB-intervalA).*rand(1,samples/2);

%x2 = intervalA + (intervalB-intervalA).*rand(1,samples/2);
%y2 = intervalA + (intervalB-intervalA).*rand(1,samples/2);

x = [x1 x2];
y = [y1 y2];

clusters = zeros(1,length(y));
% Plot samples
plot(x1,y1,'bx');
hold on;
plot(x2,y2,'rx');

% Plot centers
nCenters = 2;
centersCoor = intervalA + (intervalB-intervalA).*rand(nCenters,2);
%centersCoor = [4.2036   ,   4.0714;
%    1.2714    , 1.2176];
for n=1:nCenters
   plot(centersCoor(n,1), centersCoor(n,2), 'go');
end

repeat = 1;
while repeat==1

    % For each sample, see which center is closer
    for n=1:samples
        sampleDistance(n) = sqrt(2*intervalB^2); %maximum distance
        for m=1:nCenters
            tmp = pdist([x(n) y(n); centersCoor(m,1) centersCoor(m,2)]);
            if tmp < sampleDistance(n)
               sampleDistance(n) = tmp; % New distance
               clusters(n) = m; % Assign its cluster
            end
        end
    end
```

```
    % For each cluster, I get new centers, which are their means
    for m=1:nCenters
       repeat = 0;
       elements = clusters == m;
       indices = find(elements);
       tmpX = mean(x([indices]));
       tmpY = mean(y([indices]));
       % If we find a different center, establish the new ones and repeat
       if tmpX~=centersCoor(m,1) || tmpY~=centersCoor(m,2)
           centersCoor(m,1) = tmpX;
           centersCoor(m,2) = tmpY;
           repeat = 1;
       end
       % Plot each new center
       %plot(tmpX,tmpY,'mo');
    end

end

% Plot final centers
for n=1:nCenters
   plot(centersCoor(n,1), centersCoor(n,2), 'mo');
end
```