```
function [final_delta final_cluster_indexes] = vectorized_EMC_clustering_N(dataset,N)
%================================================================
==============
%CLUSTERS N-DIMENSIONAL VECTORS
%================================================================
==============

%***IMPORTANT NOTE REGARDING DATASET***

%This algorithm assumes that dataset has been sorted using sortrows(dataset,1:N)
%--------------------------------------------------------------------------

total_std = 0;

%calculates an approximation of the standard deviation
for i = 1 : N

  total_std = total_std + std(dataset(:,i));

endfor

s = 3*(total_std / N);

%Initializes variables ------------------------------------------------------

num_iterations = 25;
num_items = size(dataset,1);

max_change_ent = 0;
delta = 0;
prior_ent = log2(num_items)

for i = 1 : num_iterations

  counter = 1;
  index_vector = [];
  index_vector(1) = 1; %this is a dummy value

  cluster_indexes = [];
  cluster_indexes(1) = 1; %the first cluster starts at the first index
  current_index = 1;
  x_current = dataset(current_index, 1:N); %we ignore the classifier

  while((size(index_vector,1) > 0) && (size(index_vector,2) > 0))

    index_vector = find(sum((dataset(:,1:N) .- x_current).^2,2) > delta^2); %we don't need
to take the full norm, which saves a step
```

```
    temp_indexes = find(index_vector <= current_index); %these items have already
been clustered
    index_vector(temp_indexes) = [];

    if((size(index_vector,1) > 0) && (size(index_vector,2) > 0))

      counter = counter + 1;
      cluster_indexes(counter) = index_vector(1);

      x_current = dataset(index_vector(1),1:N);
      current_index = index_vector(1);

    endif

  endwhile

  num_indexes = size(cluster_indexes,2);

  temp_1 = cluster_indexes(2 : num_indexes);
  temp_2 = cluster_indexes(1 : num_indexes - 1);

  diff_vector = temp_1 .- temp_2;

  temp_divisor = sum(diff_vector);

  current_ent = vector_entropy(diff_vector/temp_divisor);
  change_ent = abs(prior_ent - current_ent);

  if(change_ent > max_change_ent)

    max_change_ent = change_ent;
    final_delta = delta;
    final_cluster_indexes = cluster_indexes;

  endif

  delta = delta + s/num_iterations;
  prior_ent = current_ent;

endfor

endfunction
```