

```

                                optimize_categories_3D
function [data_categories_array category_vec anchor_array H_final delta] =
optimize_categories_3D(data_array,scale_toggle)

    %takes in an array of 3-space vectors and returns a categorization of those
    vectors
    %scale_toggle determines whether the categorization looks for fine scale (0) or
    large scale (1) categories

    [x num_items] = size(data_array);

    %this portion of the code addresses a special case where the data isn't really
    random at all
    %it randomizes the data slightly by adding some noise, the level of which can be
    adjusted

    noise_level = .000001;

    for i = 1 : num_items

        temp = data_array{i};
        temp(1) = temp(1) + rand()*noise_level;
        temp(2) = temp(2) + rand()*noise_level;
        temp(3) = temp(3) + rand()*noise_level;
        data_array{i} = temp;

    endfor

%-----
%-----

%initializes the data_categories_array to the fully partitioned data_set
for i = 1 : num_items

    data_categories_array{i} = data_array{i};
    norm_vector(i) = norm(data_array{i});

endfor

%-----
%-----

s = 1.23*std(norm_vector(:));

symm = left_right_delta_3D(data_array);

min_entropy = (1 - symm)*log2(num_items);

```

```

                                optimize_categories_3D
max_cnt = 25;

%this is just an initialization
category_vec(1) = 0;
anchor_array = {};

%returns a measure of the symmetry of information in the data_set
if(scale_toggle == 0)

    [N s_0] = partition_array_3D(data_array); %fine structure
else

    [N s_0] = partition_array_3D_bigly(data_array); %large structure
endif

divisor = N^(N*(1-symm))/10;
increment = divisor / max_cnt;

max_ent_change = 0;
H_2 = log2(num_items);
H_final = H_2;
delta = 0;

cnt = 0;

while(cnt <= divisor && H_2 >= min_entropy)

    clear category_vec_1;
    clear category_vec_2;

    max_delta = (cnt/divisor)*s;
    [data_categories_array_1 anchor_array_1] =
generate_categories_3D(data_array,max_delta);

    max_delta = ((cnt+increment)/divisor)*s;
    [data_categories_array_2 anchor_array_2] =
generate_categories_3D(data_array,max_delta);

    [x num_categories] = size(data_categories_array_1);

    for i = 1 : num_categories

        [x temp] = size(data_categories_array_1{i}); %loads the number of items in
each category
        category_vec_1(i) = temp;

```

```

                                optimize_categories_3D
endfor

[x num_categories] = size(data_categories_array_2);

for i = 1 : num_categories

    [x temp] = size(data_categories_array_2{i}); %loads the number of items in
each category
    category_vec_2(i) = temp;

endfor

category_vec_1 = category_vec_1/num_items;
category_vec_2 = category_vec_2/num_items;

H_1 = vector_entropy(category_vec_1);
H_2 = vector_entropy(category_vec_2);

if((H_1 - H_2)^2 > max_ent_change)

    max_ent_change = (H_1 - H_2)^2;
    data_categories_array = data_categories_array_2;
    category_vec = category_vec_2;
    anchor_array = anchor_array_2;
    delta = max_delta;
    H_final = H_2;

endif

cnt = cnt + increment;

endwhile

endfunction

```