Ellis Wright Feedback sheet: Assignment # 6

This feedback sheet is meant to help you to improve your assignment reports, and to understand the grading criteria.

Part of the work of an applied mathematician is to communicate with experts that may not be mathematicians. Therefore, it is important to develop one's reporting skills. In this course, the quality of the reporting is an important element in grading.

Below, a list of important points concerning the reporting are given, and it also serves as a rubric for feedback.

1. The report should be self-explanatory, not an answer to a question. This means that you have to explain what the problem is that you are addressing. Preferably, do not just copy the assignment but explain in your own words what the assignment asks you to do. **A printed code is not a sufficient answer!**

   Your grade:

   ☐ Excellent: No need to improve
   ☐ Good: Acceptable, leaving space for improvement
   ☐ Below standard : Requires more work

2. The report should be carefully typeset, paying attention to the layout. Sloppy reporting gives a negative impression and raises questions about the correctness of the results, so one needs to pay attention on the visual quality.

   Your grade:

   ☐ Excellent: No need to improve
   ☐ Good: Acceptable, leaving space for improvement
   ☐ Below standard : Requires more work

3. Well planned graphs and plots are an integral part of good reporting. When you include a plot in the report, pay attention to the following:

   (a) Make sure that the fonts are of readable size. The default font in Matlab is not acceptable, so increase the font size, e.g., using the command `set(gca,'FontSize',15)`.

   (b) Lines and curves need to be thick enough to be readable, and markers must be large enough.

   (c) Pay attention to axes. For instance, if you plot a circle, make sure that it does not appear as an ellipse, using commands like `axis('square')` or `axis('equal')`. Also, think about the scaling of the axis so that they convey the information you want. Sometimes a logarithmic scale is more informative.

   (d) Select the graphical output carefully. Is a histogram better than a continuous curve? Discrete quantities may be better represented by dots or bars than by continuous curves etc.

(e) Consider marking the axes with a label, using commands `xlabel, ylabel`.

Your grade:

- ☐ Excellent: No need to improve
- ☐ Good: Acceptable, leaving space for improvement
- ☐ Below standard : Requires more work

4. Technical correctness. Each assignment will be checked for correctness. If you get a grade less than excellent, go and check the comments on the report.

   Your grade:

   - ☐ Excellent: No need to improve
   - ☐ Good: Acceptable, leaving space for improvement
   - ☐ Below standard : Requires more work

5. Code: Including a code, or a piece of it is recommended when applicable. Pay attention on your code: A code with no comments is below standard. Well documented code is a great asset for yourself, too!

   Your grade:

   - ☐ Excellent: No need to improve
   - ☐ Good: Acceptable, leaving space for improvement
   - ☐ Below standard : Requires more work

MATH 444 **Due Date: 05/01/2022**

Assignment 6

Ellis Wright (enw30)

1 Introduction

The goal of this assignment is to apply tree regression to reconstruct an image that has been compressed. The image was 1456 x 2592 pixels but has been compressed to 15000 pixels, representing just about 0.4% of the original image. First I will go over a toy problem set to demonstrate the algorithm works.

2 Algorithm

The image we are reconstructing is stored in a data matrix (D) storing the x and y coordinate of every pixel. We store the rgb vector (V) along with each data point.

$$\mathbb{D} \in \mathbb{R}^{2x15000}$$
$$\mathbb{D} = \{x^{(1)}, ..., x^{(15000)}\}$$
$$\mathbb{V} \in \mathbb{R}^{15000x3}$$

The algorithm involves splitting the image into tiny boxes. At each iteration, we choose a box and split it in two. We start with the whole image as the first box.

2.1 Choosing Splits

When we are given a rectangle we can choose to split it anywhere along the x or y axis. We consider all the possible values we can split as space in between consecutive pixels in both directions. For each value we need a way of measuring how well that value is to split. When we split a rectangle we look at what the average color is for both of our tinier rectangles. We add up the squared error each pixel is away from the average. We do this by adding up the total squared error the red values is plus the squared error of the blue values plus the squared value of the green values for the left box (B_L) and the right box (B_R). Here $\overline{\mathbb{V}_{BR}}$ and $\overline{\mathbb{V}_{BL}}$ are the average colors of the left and right boxes.

$$\text{MSE} = \sum_{x^{(j)} \in B_L} ||v^{(j)} - \overline{V_{B_L}}||^2 + \sum_{x^{(j)} \in B_R} ||v^{(j)} - \overline{V_{B_R}}||^2$$

We choose the split that minimizes this error, noting which direction it occurs in.

2.2 Building Tree

We start with the whole data set as the initial box. We add the data to the set of leaves. There are several ways of building the tree and we looked at two different ways. One way is to build the tree by removing the oldest leaf in the set of leaves and then splitting it and adding both smaller boxes to the set of leaves. Another way, which we investigated to see if we can get a finer image was to choose the leaf that has the most data points in it. As you will see in the later figures, both do well. After choosing the split value, you need to partition the data and the corresponding pixel values in the new nodes of the tree. The color of a leaf is the average color of the points inside of it.

3 Example

We were given a dataset with three points as shown in figure 1.

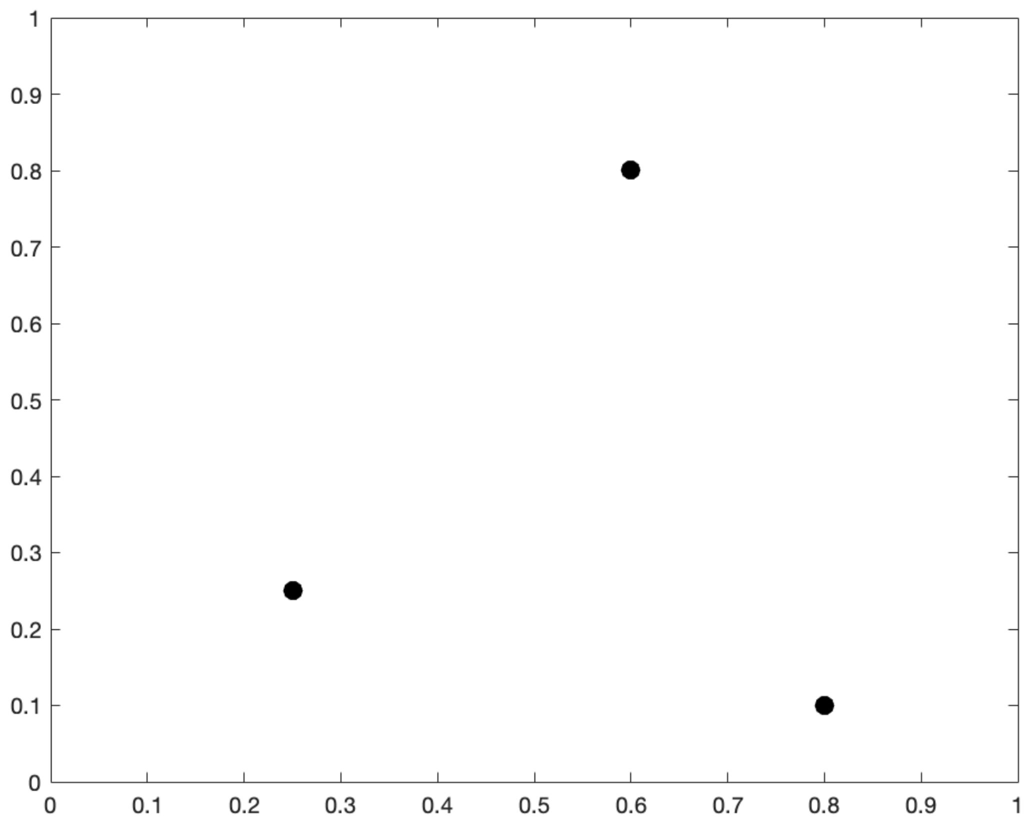
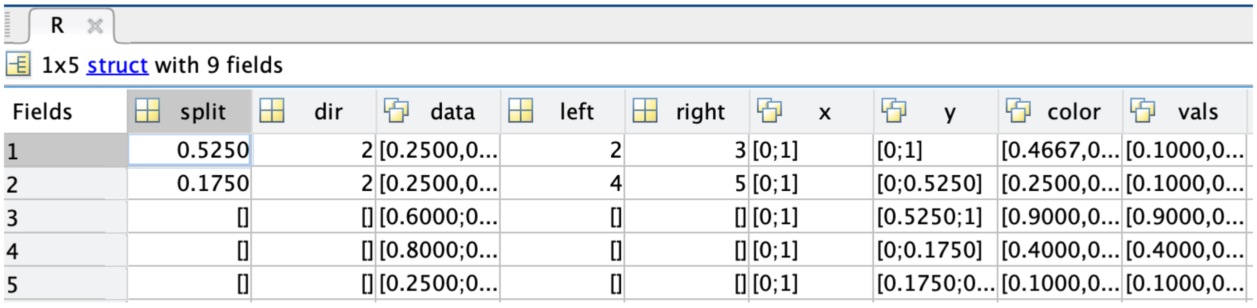


Figure 1: The three data points

Since there 3 points there are two splits in both the x and y direction. After the optimal split with the corresponding direction is found one leaf will be pure and the other will be the average of the other two. Figure 2 shows the outputs of the algorithm.



R

1x5 struct with 9 fields

Fields	split	dir	data	left	right	x	y	color	vals
1	0.5250	2	[0.2500,0...	2	3	[0;1]	[0;1]	[0.4667,0...	[0.1000,0...
2	0.1750	2	[0.2500,0...	4	5	[0;1]	[0;0.5250]	[0.2500,0...	[0.1000,0...
3	[]	[]	[0.6000;0...	[]	[]	[0;1]	[0.5250;1]	[0.9000,0...	[0.9000,0...
4	[]	[]	[0.8000;0...	[]	[]	[0;1]	[0;0.1750]	[0.4000,0...	[0.4000,0...
5	[]	[]	[0.2500;0...	[]	[]	[0;1]	[0.1750;0...	[0.1000,0...	[0.1000,0...

Figure 2: The resulting tree with children and splits

The field column is the number of the node in the tree. The dir column is the direction chosen for the split, 1 for x direction and 2 for y direction. The data stores the subsetted data in that box. Left and Right point to the children. the x and y columns are store the bounds of the boxes. Color is the average color of the vals column which stores the rgb values of the data in the box. As you can see The original node split at .52 in the y direction, then picked off the oldest leaf and chose .175 in the y direction. The second split coud have been in the x direction since both produce a pure tree but it was implemented so that the y direction was chosen in a tie. Attached at the end is my handwritten calculations of the optimal split calculations. Figure 3 shows the final image.

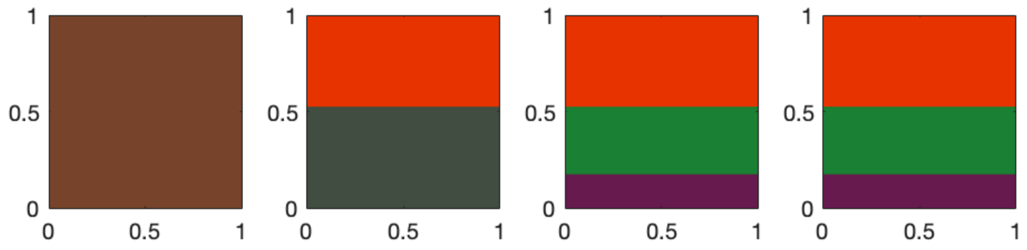


Figure 3: The final image

After two splits the tree is pure so there is no more splits in the final box.

4 Mystery Image

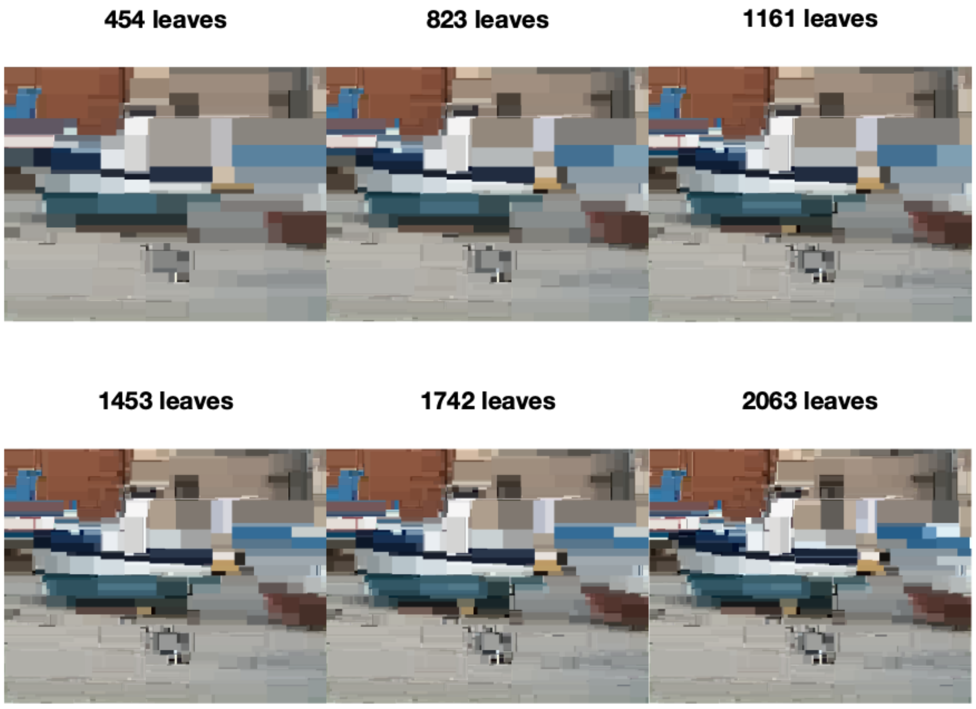


Figure 4: The mystery image throughout the algorithm

Here is the result of the mystery image. The image shows an update every 500 iterations. The reason there are less leaves is because if a pure leaf node is picked that iteration is skipped.

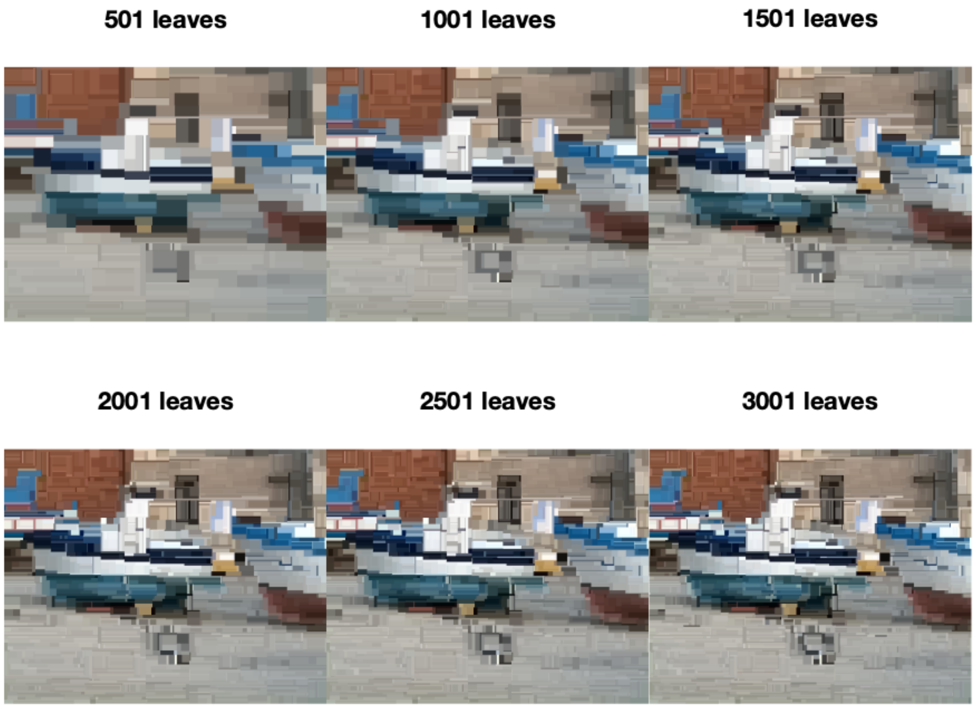


Figure 5: The mystery image throughout the algorithm choosing the node with most data to split

Using a different approach to splitting the node does in fact lead to what seems like a higher resolution image. In fact, I only tried this way because the original way looked too clunky at first. Once I made the boundary invisible for the 'fill' command in matlab, it looked a lot better but I was pleased with these results.

3412 leaves

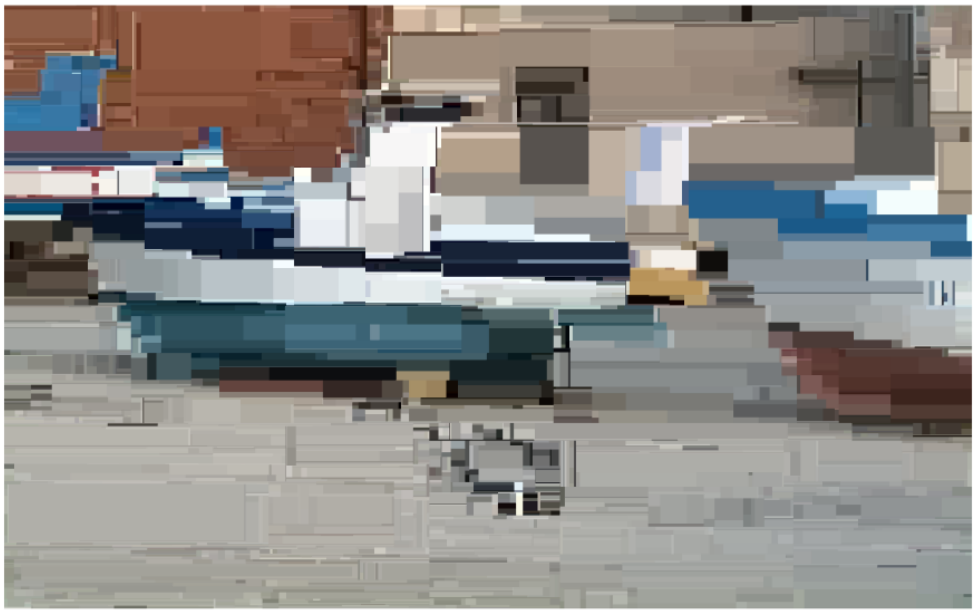


Figure 6: The final mystery image at 6000 iterations

I decided to show a hi-res photo with 6k iterations using both methods as well.

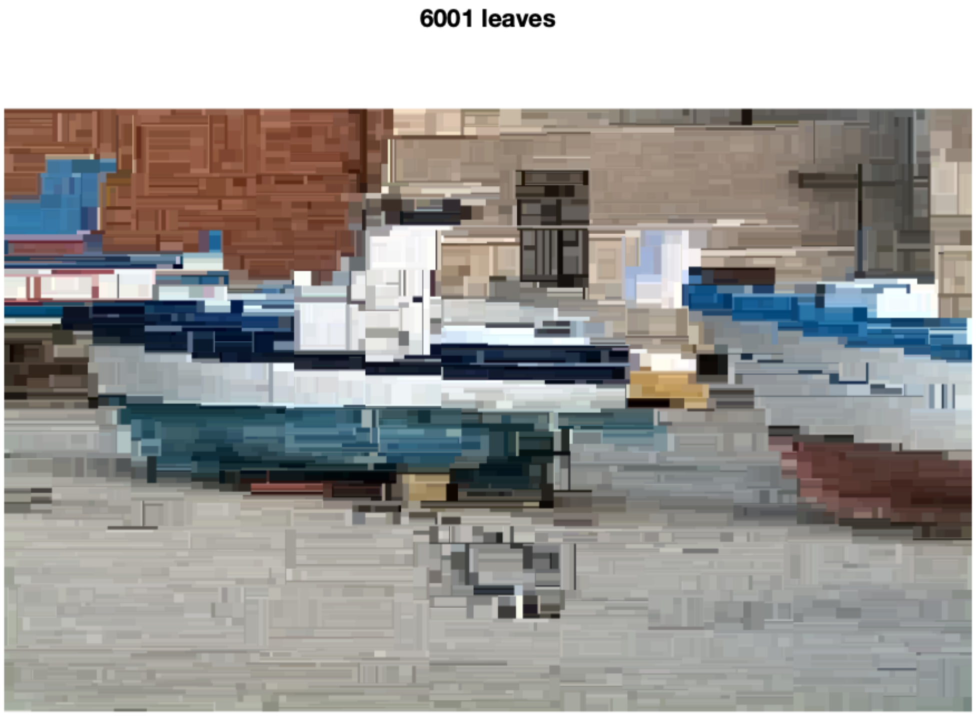


Figure 7: The final mystery image at 6000 iterations choosing node with most data

I believe the image is some kind of boat in front of a building. After consulting several other people as well, everyone had said it was some kind of boat or toy boat. I am interested in seeing the correct image!

5 Code

```
function [x_splits ,y_splits] = get_splits(x_data ,y_data)

    %get lengths of possible splits
    n = length(x_data);
    m = length(y_data);
    x_splits = zeros(1,n-1);
    y_splits = zeros(1,m-1);

    %set splits to average between points
    for i = 2:n
        x_splits(i-1) = (x_data(i) + x_data(i-1))/2;
    end

    for j = 2:m
        y_splits(j-1) = (y_data(j) + y_data(j-1))/2;
    end

end

function [direction,split_value] = opt_split(X,v)

    %split data into directions
    x = X(1,:);
    y = X(2,:);

    %get all the splits
    [x_splits,y_splits] = get_splits(unique(x),unique(y));

    %store best error and split
    bestx = nan;
    bestxerr = realmax;

    for i = x_splits
        lv = v(X(1,:) < i,:);
        rv = v(X(1,:) >= i,:);
        err = sum(sum((lv - mean(lv,1)).^2)) + sum(sum((rv - mean(rv,1)).^2));
        if err < bestxerr
            bestx = i;
            bestxerr = err;
        end
    end

    %store best error and split
    besty = nan;
    bestyerr = realmax;

    for i = y_splits
        lv = v(X(2,:) < i,:);
        rv = v(X(2,:) >= i,:);
        err = sum(sum((lv - mean(lv,1)).^2)) + sum(sum((rv - mean(rv,1)).^2));
        if err < bestyerr
            besty = i;
            bestyerr = err;
        end
    end

    %choose best split
    if bestxerr < bestyerr
        direction = 1;
        split_value = bestx;
    else
        direction = 2;
        split_value = besty;
    end

end

function plot_tree(R,leaves)

    %for each leaf plot a rectangle in its bouds
    for i = leaves
        x = R(i).x;
        y = R(i).y;
        fill([x(1),x(2),x(2),x(1)],[y(1),y(1),y(2),y(2)],R(i).color,'LineStyle','no
        hold on
    end

end

function [R,leaves] = build_tree(R,leaves)

    %t = 1;
    %besg = 0;
    %for j = 1:length(leaves)
    %    if length(R(leaves(j)).data(1,:)) > besg
    %        besg = length(R(leaves(j)).data(1,:));
    %        t = j;
    %    end
    %end
    %i = leaves(t);
    %leaves(t) = [];

    %get oldest leaf
    i = leaves(1);
    leaves(1) = [];

    %if pure, return
    if length(R(i).data(1,:)) == 1
        leaves = [leaves,i];
        return
    end

    %get optimal split and direction
    [R(i).dir,R(i).split] = opt_split(R(i).data,R(i).vals);

    %subset indiceis
    Il = R(i).data(R(i).dir,:) < R(i).split;
    Ir = R(i).data(R(i).dir,:) >= R(i).split;

  %subset data
    left = R(i).data(:,Il);
    right = R(i).data(:,Ir);

    %subset pixel values
    lvals = R(i).vals(Il,:);
    rvals = R(i).vals(Ir,:);

    l = length(R);

    %initialize nodes
    R(i).left = l + 1;
    R(i).right = l + 2;

    x = R(i).x;
    y = R(i).y;

        R(l+1).split = [];
        R(l+1).dir = [];
        R(l+1).data = left;
        R(l+1).left = [];
        R(l+1).right = [];
        if R(i).dir == 1
            R(l+1).x = [x(1);R(i).split];
            R(l+1).y = y;
        else
            R(l+1).x = x;
            R(l+1).y = [y(1);R(i).split];
        end

        R(l+1).color = mean(lvals,1);
        R(l+1).vals = lvals;


        R(l+2).split = [];
        R(l+2).dir = [];
        R(l+2).data = right;
        R(l+2).left = [];
        R(l+2).right = [];

        if R(i).dir == 1
            R(l+2).x = [R(i).split;x(2)];
            R(l+2).y = y;
        else
            R(l+2).x = x;
            R(l+2).y = [R(i).split;y(2)];
        end

        R(l+2).color = mean(rvals,1);
        R(l+2).vals = rvals;

        leaves = [leaves,l+1,l+2];
end
```

CODE THAT GENERATED TOY IMAGES

```
x1 = 0.25;
y1 = 0.25;
v1 = [0.1, 0.5, 0.2];
x2 = 0.8;
y2 = 0.1;
v2 = [0.4, 0.1, 0.3];
x3 = 0.6;
y3 = 0.8;
v3 = [0.9, 0.2, 0.0];

m = 1;
n = 1;

X = [[x1,x2,x3];[y1,y2,y3]];
v = [v1;v2;v3];

R(1).split = [];
R(1).dir = [];
R(1).data = X;
R(1).left = [];
R(1).right = [];
R(1).x = [0;1];
R(1).y = [0;1];
```

```
R(1).color = 1/length(v(:,1))*sum(v,1);
R(1).vals = v;

[i_opt,s_opt] = find_best_split(X,v);

[x_splits,y_splits] = get_splits(unique(X(1,:)),unique(X(2,:)));

bestx = nan;
bestxerr = realmax;

for i = x_splits
    lv = v(X(1,:) < i,:);
    rv = v(X(1,:) >= i,:);
    err = sum(sum((lv - mean(lv,1)).^2)) + sum(sum((rv - mean(rv,1)).^2));
    if err < bestxerr
        bestx = i;
        bestxerr = err;
    end
end

besty = nan;
bestyerr = realmax;

for i = y_splits
    lv = v(X(2,:) < i,:);
    rv = v(X(2,:) >= i,:);
    err = sum(sum((lv - mean(lv,1)).^2)) + sum(sum((rv - mean(rv,1)).^2));
    if err < bestyerr
        besty = i;
        bestyerr = err;
    end
end

if bestxerr < bestyerr
    dir = 1;
    split = bestx;
else
    dir = 2;
    split = besty;
end

leaves = (1);
tiledlayout(1,4)
nexttile
plot_tree(R,leaves)
```

```
axis square
nexttile
[R,leaves] = build_tree(R,leaves);
plot_tree(R,leaves)
axis square
nexttile
[R,leaves] = build_tree(R,leaves);
plot_tree(R,leaves)
axis square
nexttile
[R,leaves] = build_tree(R,leaves);
plot_tree(R,leaves)
axis square
```

CODE THAT GENERATED MYSTERY IMAGE

```
load('MysteryImage.mat')
m = 1456;
n = 2592;

x_data = cols/n;
y_data = (m/n)*(1 - rows/n);

data = [x_data';y_data'];

R(1).split = [];
R(1).dir = [];
R(1).data = data;
R(1).left = [];
R(1).right = [];
R(1).x = [0;1];
R(1).y = [.246;m/n];
R(1).color = mean(vals,1);
R(1).vals = vals;

leaves = (1);

figure(1)
tiledlayout(2,3,TileSpacing='none')
for i = 1:3000
    [R,leaves] = build_tree(R,leaves);
    if mod(i,500) == 0
        nexttile
        plot_tree(R,leaves)
        axis square
        axis off
        title(string(length(leaves)) + " leaves")
    end
end

figure(2)
plot_tree(R,leaves)
title(string(length(leaves)+" leaves"))
axis off
```

$$X = \begin{pmatrix} .25 & .8 & .6 \\ .25 & .1 & .8 \end{pmatrix} \qquad V = \begin{pmatrix} .1 & .5 & .2 \\ .4 & .1 & .3 \\ .9 & .2 & 0 \end{pmatrix}$$

x-splits = (.425 , .7)

y-splits = (.175 , .525)

$\underline{.425)}$

$x(1,:) < .425 \quad = (1\ 0\ 0) \quad \Rightarrow \text{lerr} = 0$

$x(1,:) > .425 \quad = (0\ 1\ 1) \quad \rightarrow \text{solve}$

$$v([0\,1\,1],:) = \begin{pmatrix} .4 & .1 & .3 \\ .9 & .2 & 0 \end{pmatrix} \quad (v-\bar{v})^2 = \begin{pmatrix} (-.25)^2 & (-.05)^2 & (.15)^2 \\ (.25)^2 & (.05)^2 & (.15)^2 \end{pmatrix}$$

mean = (.65 .15 .15)

$$+ \quad \overline{}$$

$$0.175$$

lerr < rerr so current best is (1, .425)

$\underline{.7)}$

$x(1,:) < .7 \quad = (1\,0\,1) \quad \rightarrow \text{solve}$

$x(1,:) > .7 \quad = (0\,1\,0) \quad \Rightarrow \text{rerr} = 0$

$$V(C101), :) = \begin{pmatrix} .1 & .5 & .2 \\ .4 & .2 & .0 \end{pmatrix} \quad (v-\bar{v})^2 = \begin{pmatrix} (-.4)^2 & (.15)^2 & (.1)^2 \\ (.4)^2 & (-.15)^2 & (-.1)^2 \end{pmatrix}$$

$$\text{mean} = (.5 \quad .35 \quad .1) \qquad + \overline{\qquad\qquad\qquad}$$

$$.385$$

$$r_{err} < l_{err}$$

$$\text{bestxsplit} = (1, \quad .425) \overset{xerr}{(.175)}$$

y-split

$$X(2,:) < .175 = (0\,1\,0) \Rightarrow l_{err} = 0$$

$$X(2,:) \geq .175 = (1\,0\,1) \rightarrow \text{solve} = .385$$

$$X(2,:) < .525 = (1\,1\,0) = \text{solve}$$

$$X(2,:) \geq .525 = (0\,0\,1) \Rightarrow r_{err} = 0$$

$$v(110), :) = \begin{pmatrix} .1 & .5 & .2 \\ .4 & .1 & .3 \end{pmatrix} \quad (v-\bar{v})^2 = \begin{pmatrix} (.15)^2 & (.2)^2 & (.15)^2 \\ (.15)^2 & (.2)^2 & (.15)^2 \end{pmatrix}$$

mean(.25 .3 .25) +

.17

$\text{bestysplit}(.525, \overset{\text{yerr}}{.17})$

yerr < xerr so split at y = .525

create rectangle

$$R(2) = \begin{pmatrix} .25 & .8 \\ .25 & .1 \end{pmatrix} \qquad R(3) = \begin{pmatrix} .6 \\ .8 \end{pmatrix}$$

$$\text{vals} \begin{pmatrix} .1 & .5 & .2 \\ .4 & .1 & .3 \end{pmatrix} \qquad \text{vals} \ (.9 \ .3 \ .0)$$

(pure)

xsplits = (.525)

ysplits = (.175)

$$\left((.1 \quad .5 \quad .2) - (.1 \quad .5 \quad .2)\right)^2$$

$$\begin{matrix} 1 \\ 1 \\ 1 \end{matrix}$$

$\textcircled{0}$

xent