Fashion Recommendation System

Approach:

1) Importing Model (RESNET)

2) Extracting features from images

3) Exporting features in form of pickle file

4) Generating Recommendation for the query point.

Plan of Attack:

1) We have Around 44K+ images.

Reshaping it to (224, 224, 3) using (PIL.Image).

2) Now, Using RESNET. model, We'll predict 2048 features for each image.

3) During testing time, We'll generate 2048 features for that image & using KNN or other ML algorithm, I'll try to show 5 most related images to the query point.

Flow Diagram:

AAK Images

↓ Reshaping

(224, 224, 3)

↓ Extracting 2048 most prominent features

(7, 7, 2048)

↓ Global max pooling

(1, 1, 2048).

--- 2048 dimensions

---→ Each image a vector in 2048 dimensions..

For Query point —

Image

↓ Reshaping

(224, 224, 3)

↓ Extracting Features

(7, 7, 2048)

↓ Global Max Pooling

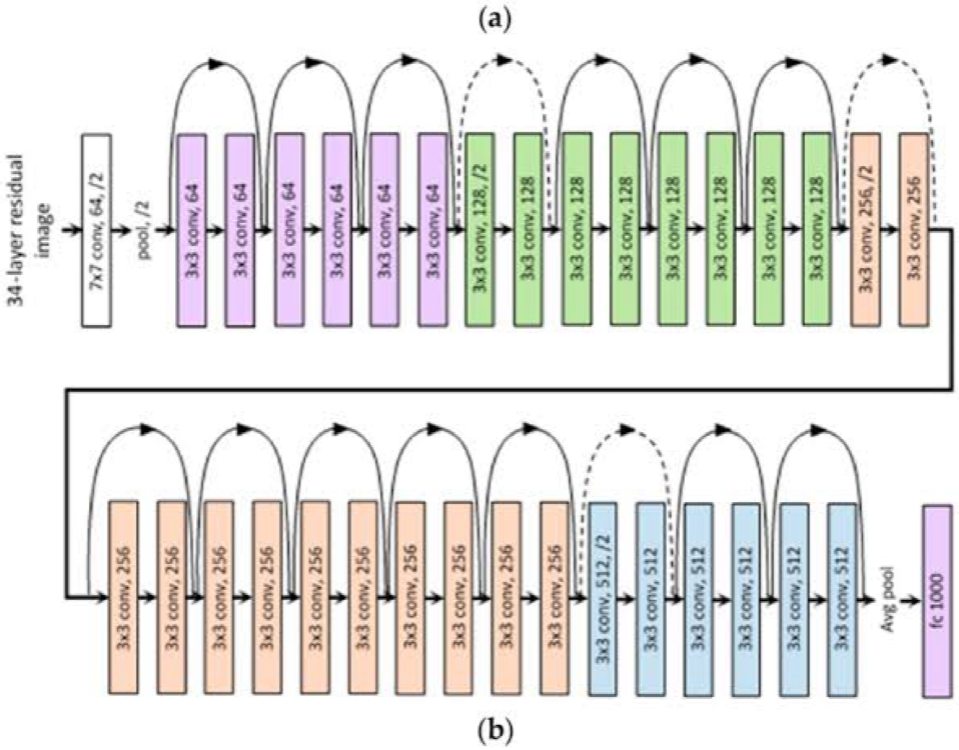
(1, 1, 2048)

Now, we'll compute the distance of this vector from existing 44K+ vectors in 2048 dimensional space.

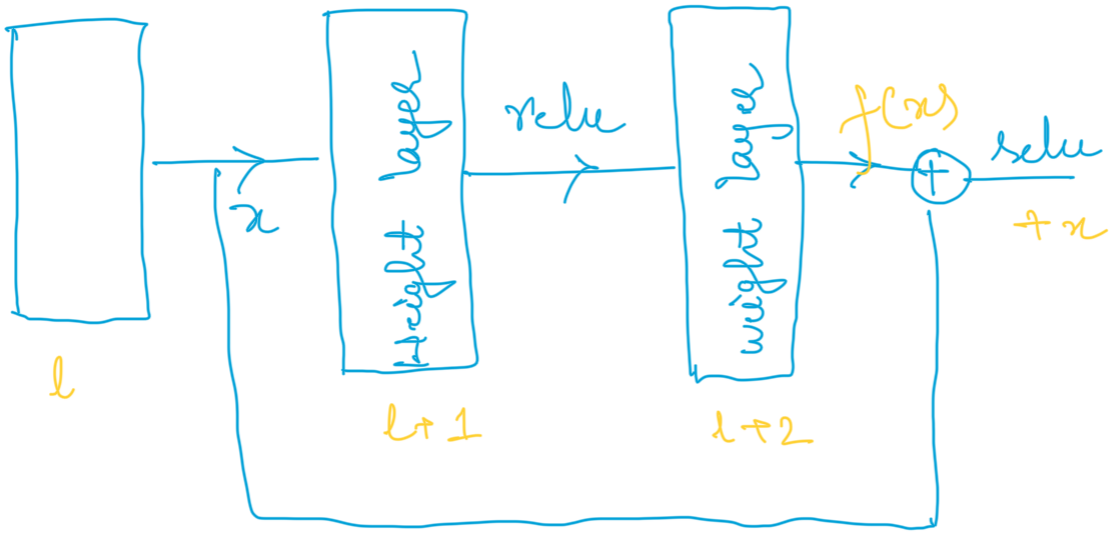
↓

Display the 5 most closest movies

RESNET ARCHITECTURE



RESIDUAL BLOCK



$$a^{[l]} = x$$

$$Z^{[l+1]} = W^{[l+1]} \cdot a^{[l]} + b^{[l+1]}$$

$$a^{[l+1]} = relu\left(Z^{[l+1]}\right)$$

$$Z^{[l+2]} = W^{[l+2]} \cdot a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+2]} = relu\left(W^{[l+2]} \cdot a^{[l+1]} + b^{[l+2]} + a^{[l]}\right)$$

1) CONCEPT OF RESIDUAL BLOCK HELPS IN mitigating overfitting....

2) Acts as a Regularizer...

$y = f(x) + x$

When

$f(x) \to 0$

$y = x$

Providing multiple path for gradient flow - ↓ Prevents reliance on single path

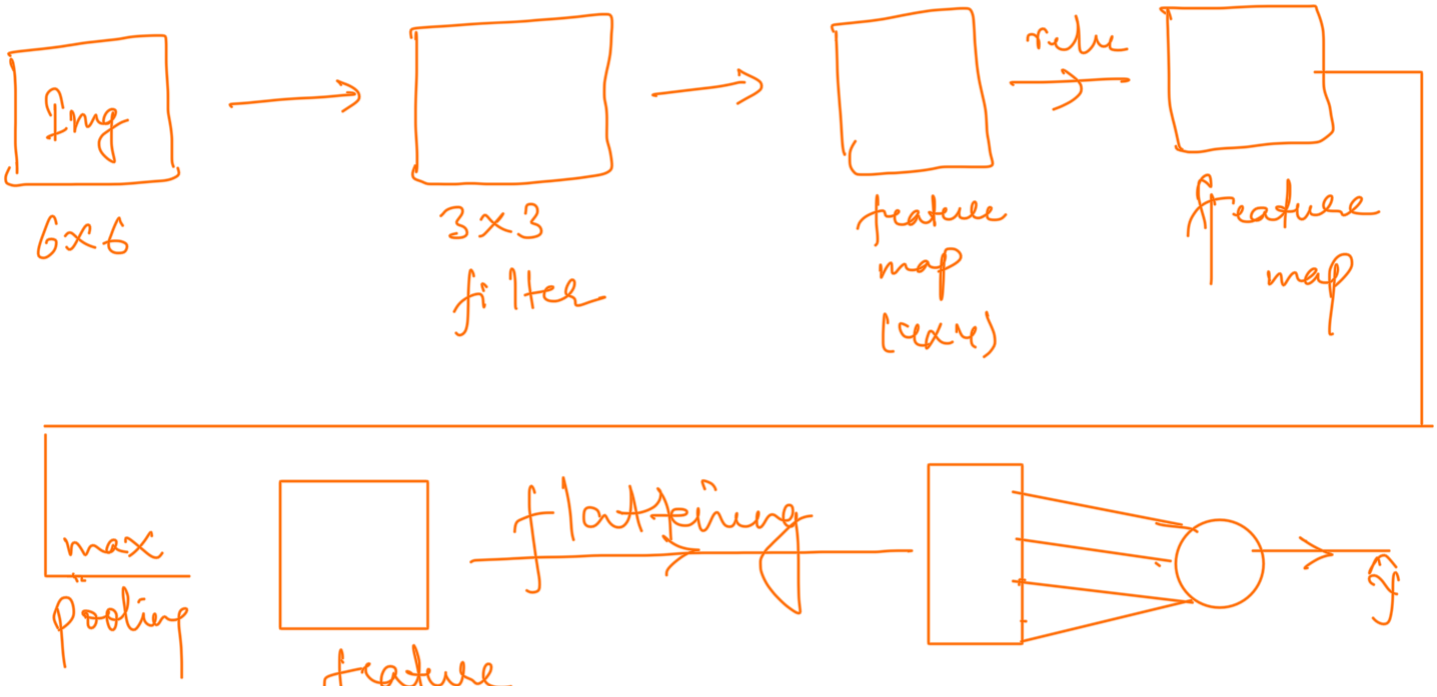
☆ during backpropagation, we usually face Vanishing Gradient or Exploding Gradient type problem...

Through Residual block, we have shorter connections which allow gradients to flow more directly through the network...

☆☆

WHY DO WE FACE PROBLEM OF VANISHING GRADIENT

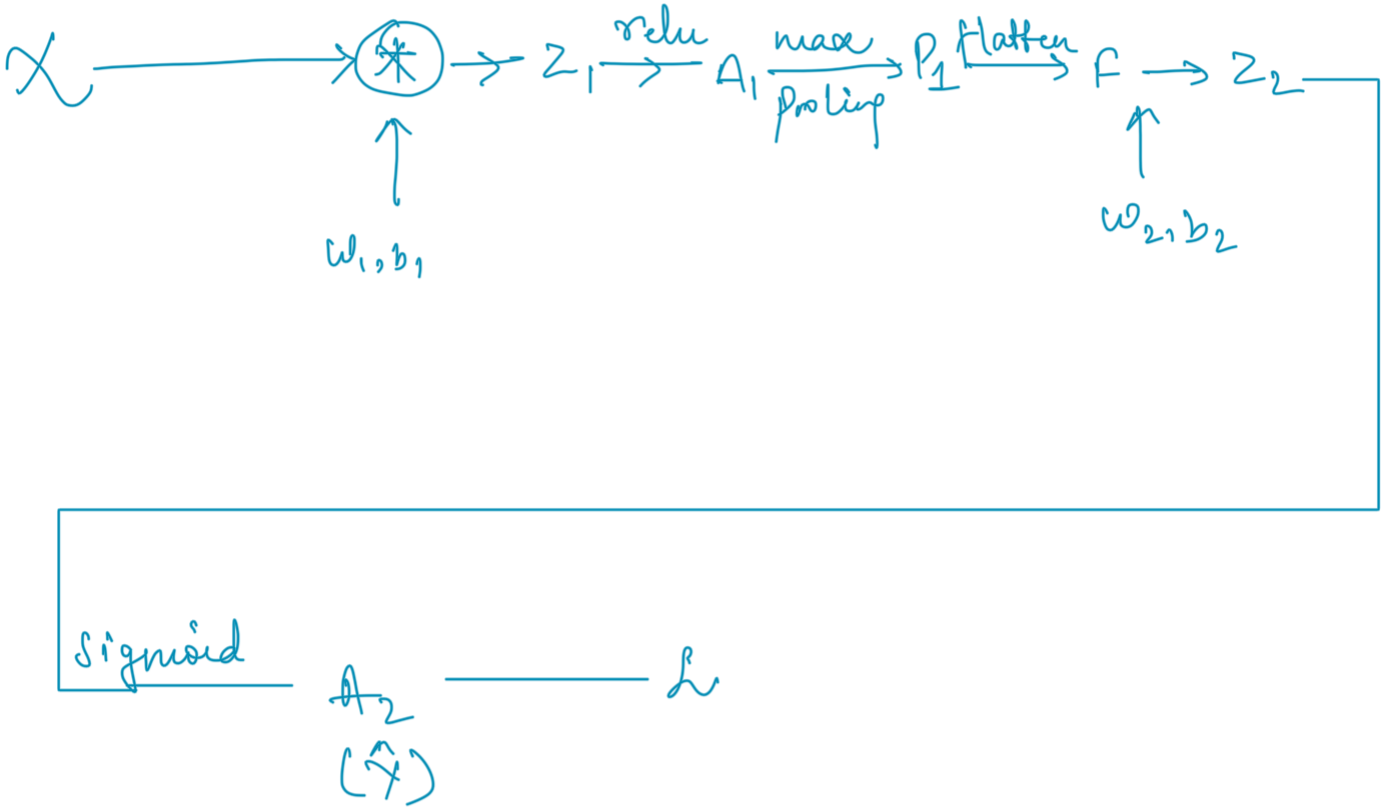
Let's assume a simple architecture



(2×2)
stride = 2

map

Forward Propagation ...



Dealing with classification problem (Binary)

so we'll use Binary Cross Entropy.

Forward Propagation eqns.

$$Z_1 = \text{conv}(x, w_1) + b$$

$$A_1 = \text{relu}(Z_1)$$

$$P_1 = \text{maxpool}(A_1)$$

$$f = \text{flatten}(P_1)$$

$$Z_2 = w_2 \cdot f + b_2$$

$$A_2 = \sigma(Z_2)$$

Loss fn $= -y_i \log(\hat{y_i}) - (1-y_i) \log(1-\hat{y_i})$

Objective fn $= \frac{1}{n} \sum_{i=1}^{n} -y_i \log(\hat{y_i}) - (1-y_i) \log(1-\hat{y_i})$

Back Propagation eqns.

$$w_1 = w_1 - \eta \frac{\partial L}{\partial}$$

$$b_1 = b_1 - \eta \frac{\partial L}{\partial b_1} \quad \downarrow \partial w_1$$

$$w_2 = w_2 - \eta \frac{\partial L}{\partial w_2}$$

$$b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_2}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial b_2}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial P} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial A_2} \times \frac{\partial A_2}{\partial z_2} \times \frac{\partial z_2}{\partial P} \times \frac{\partial P_1}{\partial A_1} \times \frac{\partial A_1}{\partial z_1} \times \frac{\partial z_1}{\partial b_1}$$

Observations

↳ just for one layer, in order to back propogate error, we have to calculate such a large no. of gradients

Imagine, more than 20 layer, you could think of how complex will it be to propogate error through backpropogation.

If the value of gradient $\longrightarrow 0$ ↓

Multiplication of gradient will be nearly 0.

$$W_1 = W_1 - \eta \left(\frac{\partial L}{\partial \cdot}\right) \longrightarrow 0$$

No Training..

But in RESNET, THROUGH RESIDUAL BLOCKS, GRADIENT FLOW IS MONITORED WELL.....

In order to make web app scalable, I am using ANNOY (Spotify) technique to compute distance b/w the vectors...

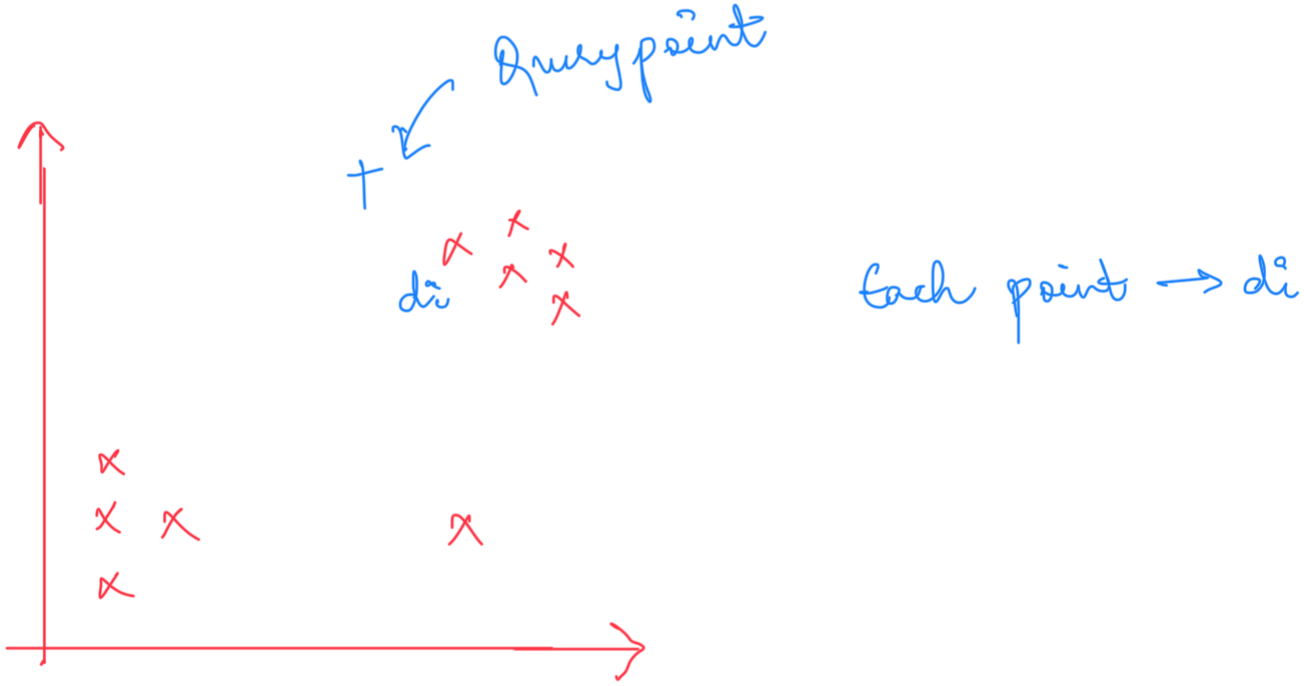
KNN (K-Nearest Neighbour)

Suppose,

$K = 3$

Approach : Brute

Method = Euclidean :



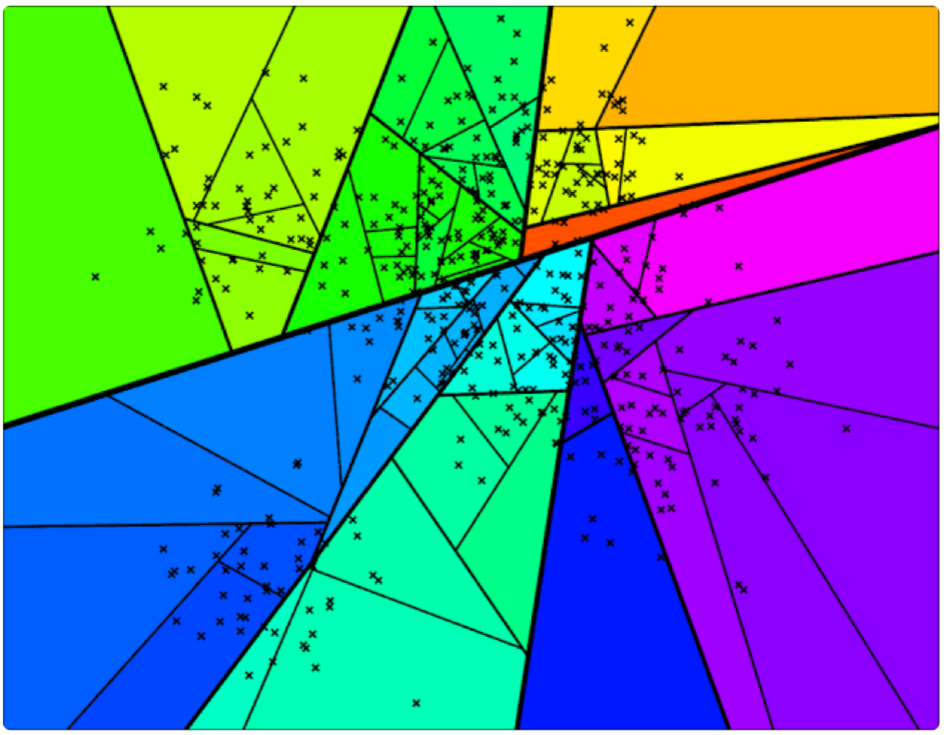
i	0	1	2
	d_0	d_1	d_2

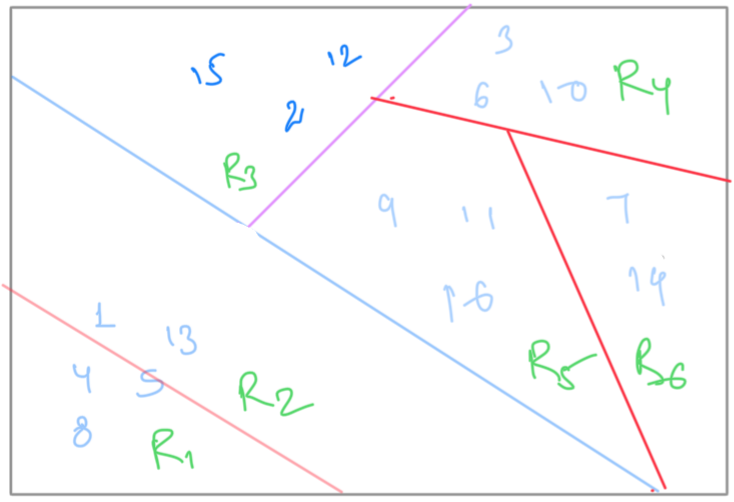
(Min Distance Table)

☆ We take a point & compute its distance from rest n points & populate the min distance table & in the end we'll have nearest vector values... (index)

Rough time complexity = $O(n)$

ANNOY (Approximate Nearest Neighbours oh yeah)





ANNOY.

Assumptions

→ We won't split in the region if it contains less than threshold point.

Here, Threshold value = 3.

Earlier in case of KNN, we used to compute distance of a point from rest all in points in a vector space.

But in case of ANNOY, we just pick the value of its region & thus a lot of computational cost is saved..

KNN	ANNOY
$O(n)$	$O(n/2)$

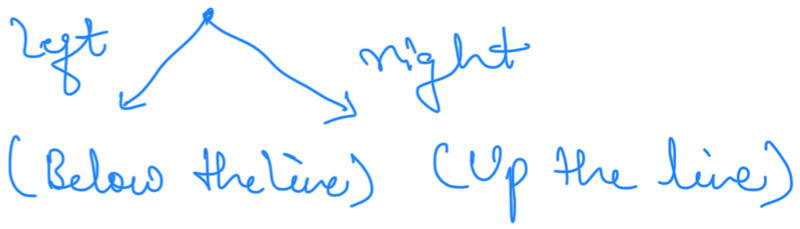
Example of How splitting is carried out?

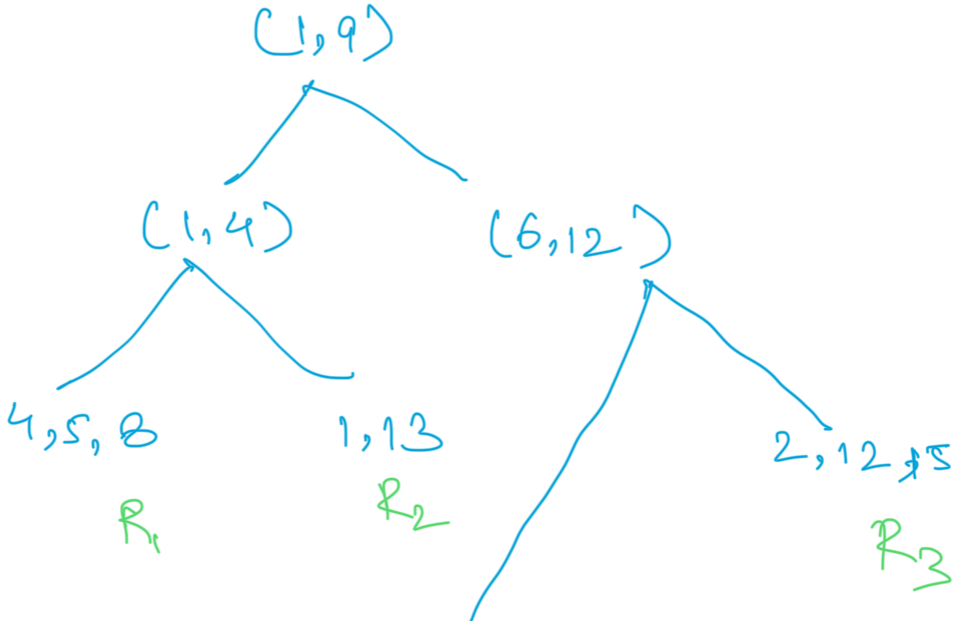
☆ Picking two Random points & finding

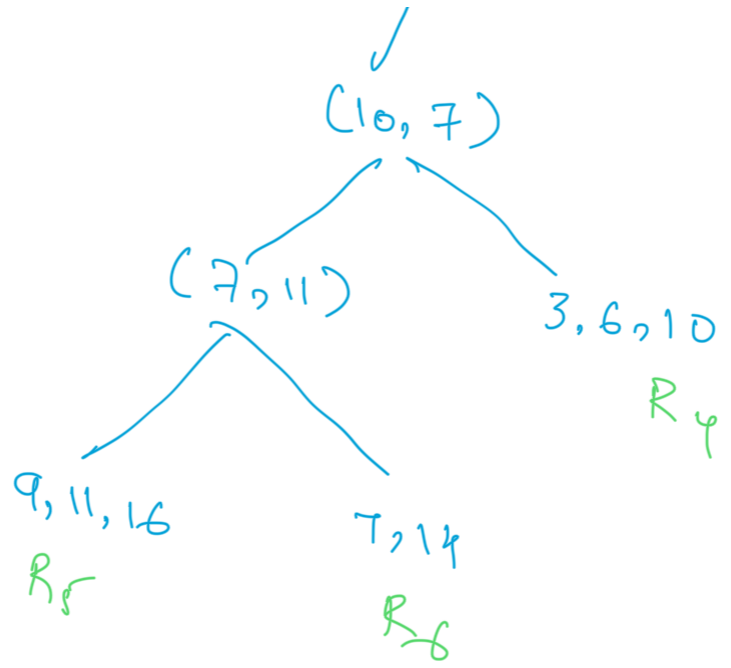
a plane perpendicular to both points & equidistant.

☆ Taking the convention ↓

In our case, let it be as follows!







☆ ☆
☆

Note: I have explained the simplistic approach of ANNOY to understand the fastness it provides...

THANKING YOU!!

Abhishek Singh
IIIT BH CSE Department