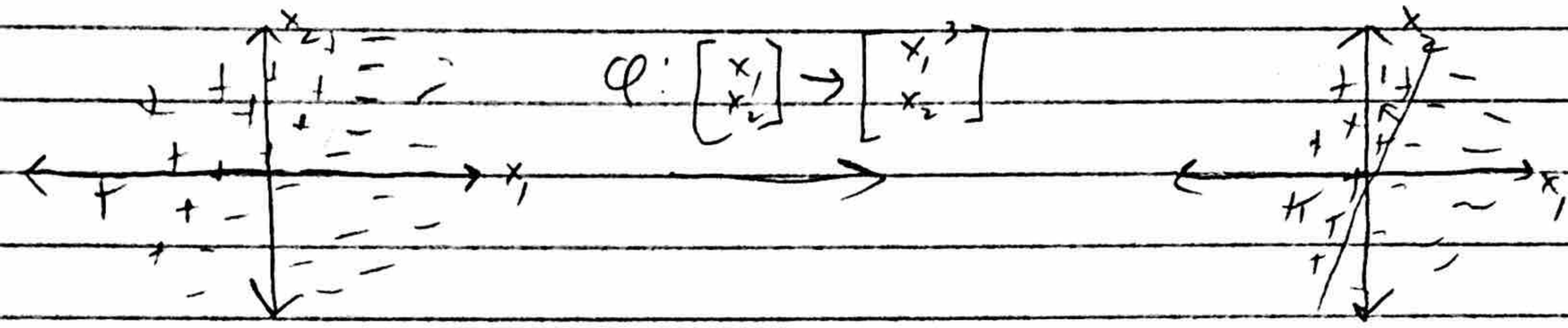
Lecture 5- Ensembles and the Random Forest Algorithm

Review

① What is the non-linear data transformation ϕ which would transform the following dataset to a linearly separable dataset?



② Challenge: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$, $k(x,z) = (x \cdot z + 1)^2$

What is the data transformation function ϕ which the kernel computes?
i.e. what is ϕ such that $k(x,z) = \phi(x) \cdot \phi(z)$?

$$k(x,z) = (x \cdot z + 1)^2 = (x_1 z_1 + x_2 z_2 + 1)^2 = x_1^2 z_1^2 + x_2^2 z_2^2 + 1^2 + 2 x_1 z_1 x_2 z_2 + 2 x_1 z_1 + 2 x_2 z_2 =$$

$$\begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2} x_1 \\ \sqrt{2} x_2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2} z_1 z_2 \\ \sqrt{2} z_1 \\ \sqrt{2} z_2 \\ 1 \end{pmatrix}$$

$$\therefore \phi(x) = \phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2} x_1 \\ \sqrt{2} x_2 \\ 1 \end{pmatrix}$$

Looking forward

A single classifier can be biased by its initialization or by certain aspects of the data. Can we build a better model by learning multiple classifiers? How?

5-2

Today

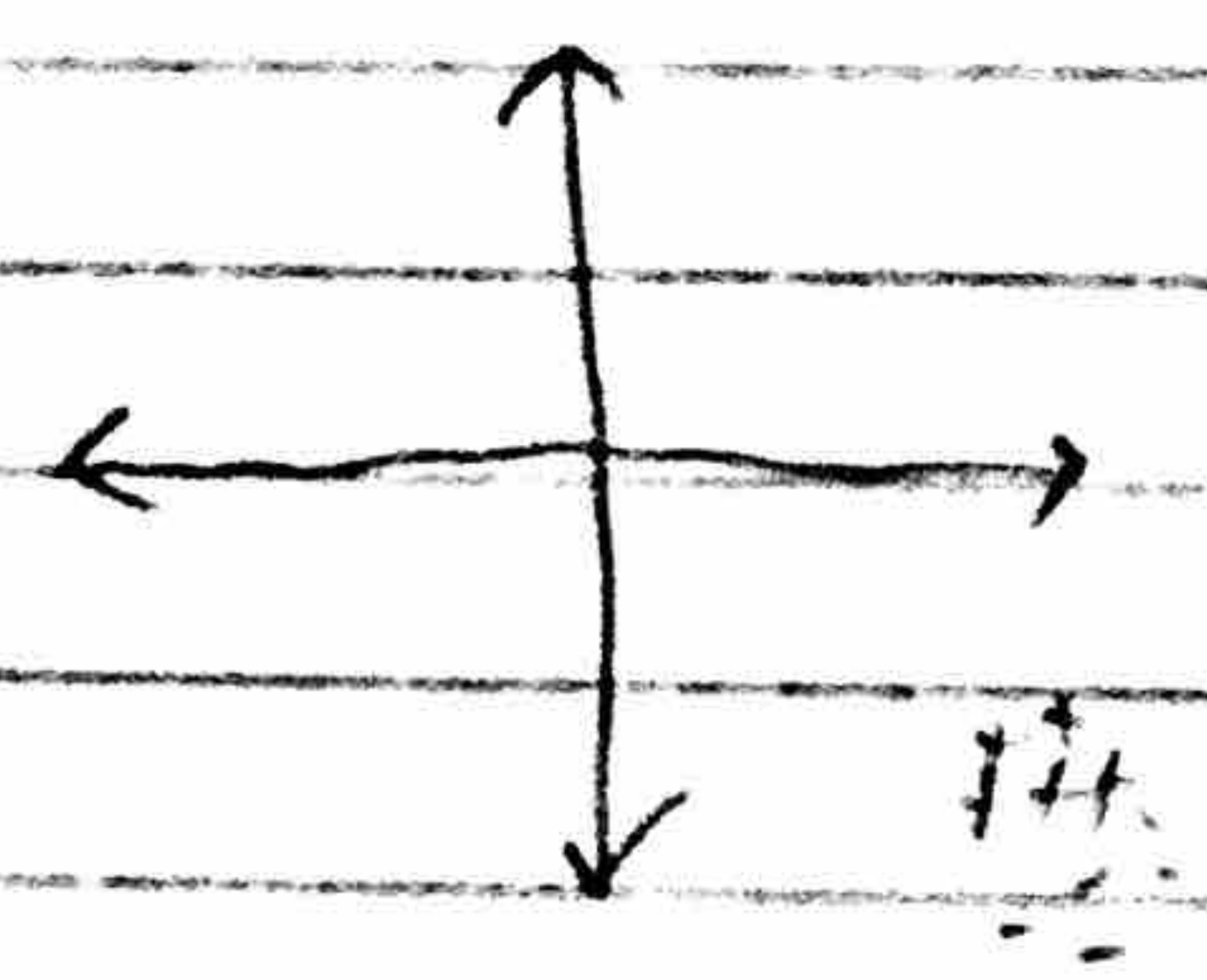
- Building a model from multiple classifiers
- Random forest algorithm
  - Decision tree
  - Decision forest
  - Random forest

Ensembles

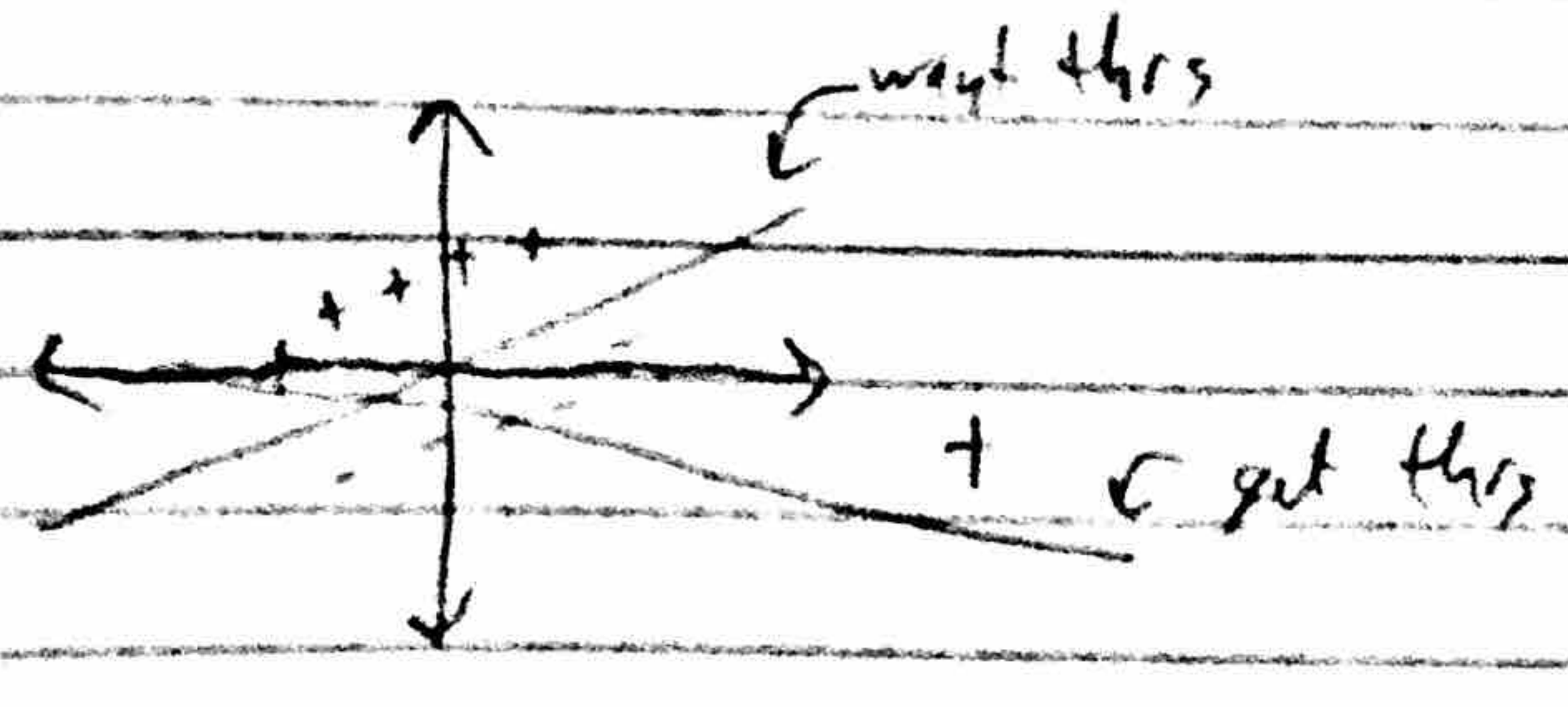
Building a model from multiple classifiers

(?) How? Individual classifiers can be biased'

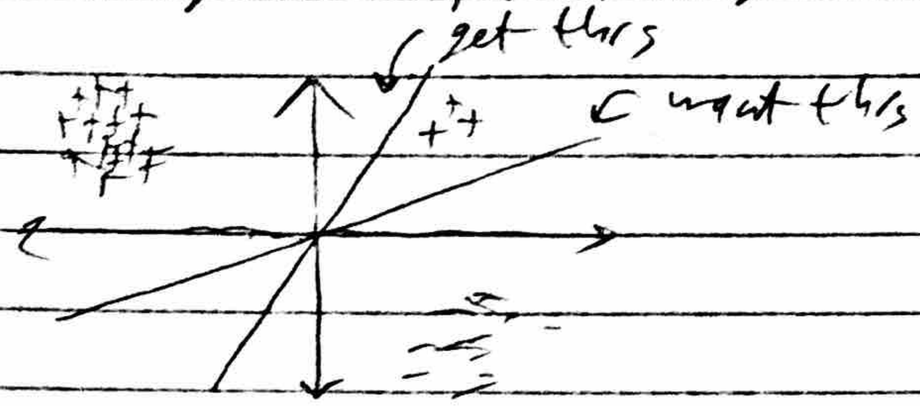
- Initialization
  - Ex. θ, θ_0 in perceptron. Maybe 0 is a bad first guess and will make it harder to learn the true θ, θ_0.



- Outliers



- Majority outweighs minority



An ensemble of classifiers can help reduce bias.

How to build an ensemble model

Train multiple classifiers with different initializations and/or subsets of the data

Now few classifiers will include the outliers.
Now some classifiers may give more weight to the minority of the data.
So we can combine the power of classifiers which do well on the majority and classifiers which do well on the minority to get a model which works well for both.

How to predict with an ensemble model

- Equal voting
  - Output the label predicted by a majority of the classifiers

Ex. If we have classifiers $h_1, h_2, \ldots, h_m$ which predict +1 or −1, then our ensemble classifier is: $\boxed{h(x) = \text{sign}(h_1(x) + h_2(x) + \ldots + h_m(x))}$

- Weighted voting
  - Prioritize the predictions of the classifiers (which predict +1 or −1 which perform better)

Ex. If we have classifiers $h_1, h_2, \ldots, h_m$ with validation accuracies $a_1, a_2, \ldots, a_m$, then our ensemble classifier is:

$\boxed{h(x) = \text{sign}(a_1 \cdot h_1(x) + a_2 \cdot h_2(x) + \ldots + a_m \cdot h_m(x))}$

Random Forest Algorithm
We can build ensembles of any type of classifier, but the random forest algorithm is a commonly used ensemble algorithm

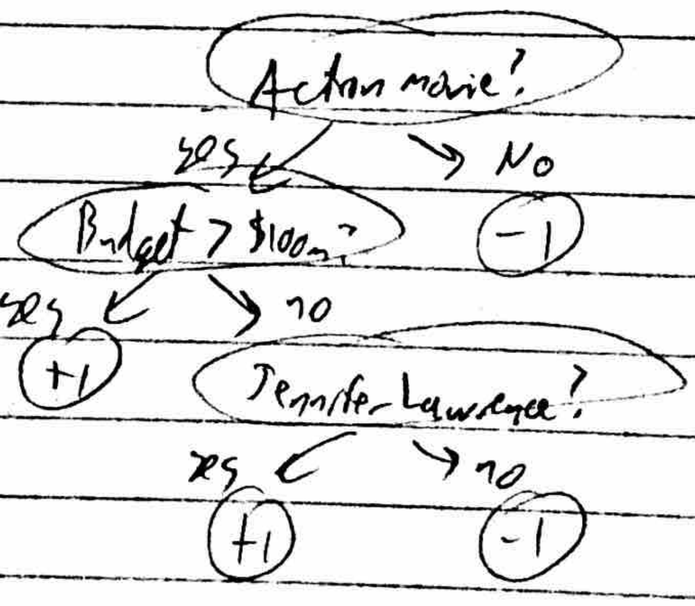
Decision Trees
Idea: Build a flowchart of rules to make a prediction

Example: Let's say we want to predict whether a movie gets a good rating.
Input features: Genre, Cast, budget, sequel
Output: +1 or -1

(?) Ask for features

Manually constructed decision tree

(?) What do you think is most important feature? Next? etc.



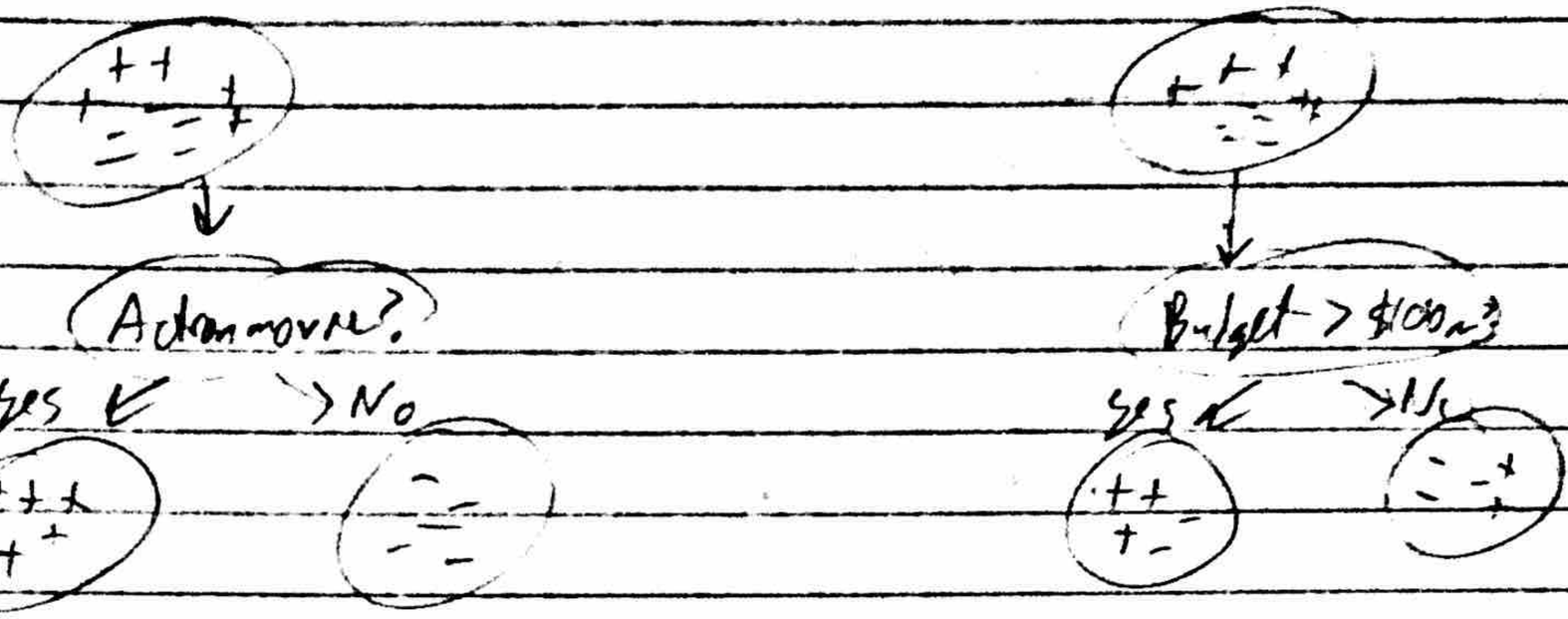
(?)

Learned decision tree
How can we build a model which learns which rules to use from the data?

Idea: At each level of the tree, use the rule that provides the best split of the data.

→

(1) Which rules is better?



(2) How do we quantify how good the split is?

we will only cover gini because it's simpler

Two methods: gini index and information gain/entropy

The gini impurity index is a measure of how well separated (pure) the data is.

$G = 1 - \sum_{i=1}^{n_c} (p_i)^2$, $G = 1 - (p_+^2 + p_-^2)$ $\quad p_+ = $ proportion of + $\quad p_- = $ proportion of - $\quad \in [0, 1]$

n_c = # classes

2 class

Impure: $p_+ = \frac{1}{2}, p_- = \frac{1}{2} \rightarrow G = \frac{1}{2}(1 - \frac{1}{2}) + \frac{1}{2}(1 - \frac{1}{2}) = \frac{1}{2}$

Pure: $p_+ = 1, p_- = 0 \rightarrow G = 1(1-0) + 0(1-1) = 0$

$G \in [0, \frac{1}{2}]$

Small G = pure = good split

Large G = impure = bad split

Goal: minimize gini index after split (minimum impurity = best split)

$$G_{split} = \frac{n_{yes}}{n} \cdot G_{yes} + \frac{n_{no}}{n} \cdot G_{no}$$

where n = # both points before split

n_{yes} = # in yes category

n_{no} = # in no category

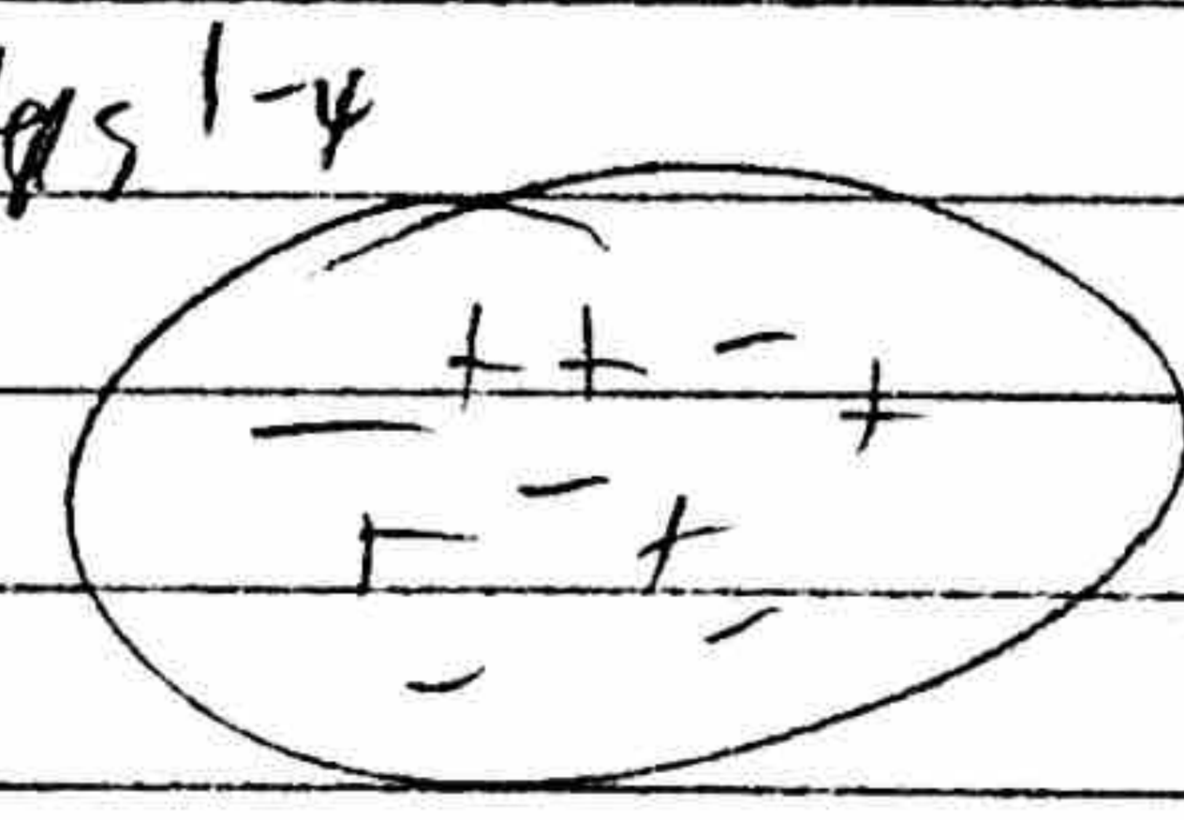
G_{yes} = gini index of yes category

G_{no} = gini index of no category

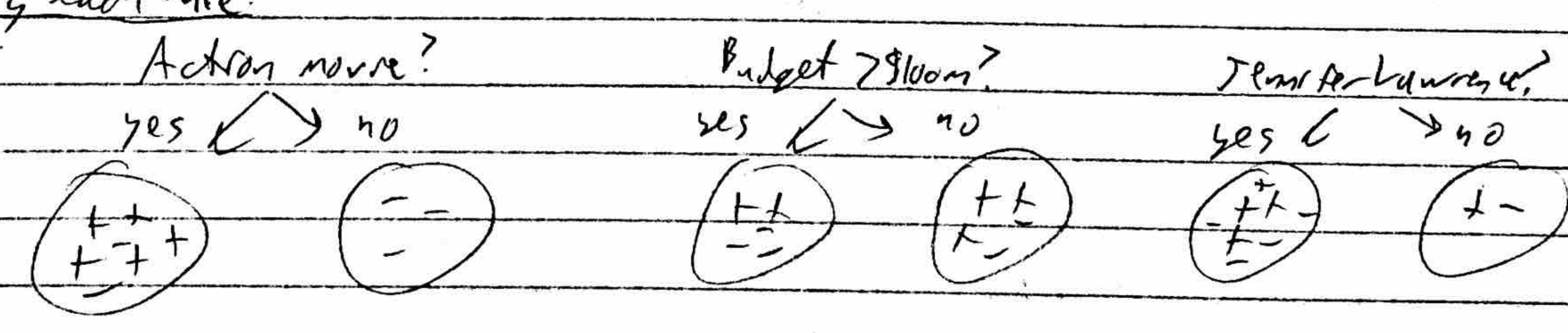
Decision Tree Algorithm

1. For each unused rule:
   a) Determine the split of the data
   (i.e. which data points are in the yes category and which are in the no category)
   b) Compute the gini index of the split
2. Select the rule with the lowest gini index
3. Add a node to the tree for that rule, along with two child nodes
   one for yes and one for no,
4. For each childnode:
   if gini index = 0:
   add label node and stop
   else:
   repeat steps 1-4

Example



Try each rule:



(?) Which one do you think is best?

(?) Compute gini of action

$$G = \frac{7}{10}\left(1-\left(\left(\frac{5}{7}\right)^2+\left(\frac{2}{7}\right)^2\right)\right) + \frac{3}{7}\left(1-(1^2+0^2)\right)$$

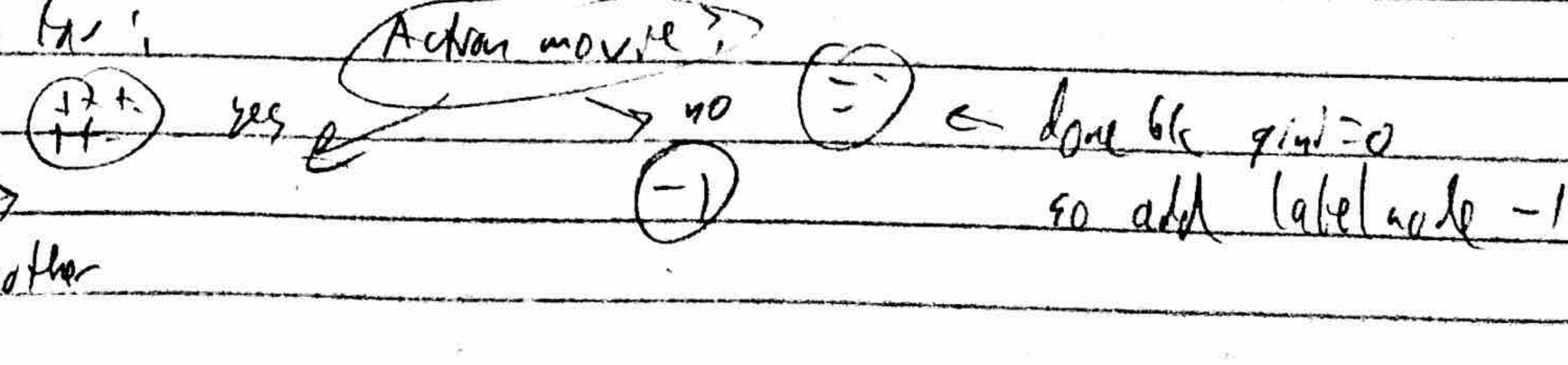
$= 0.29$

$G = 0.46$

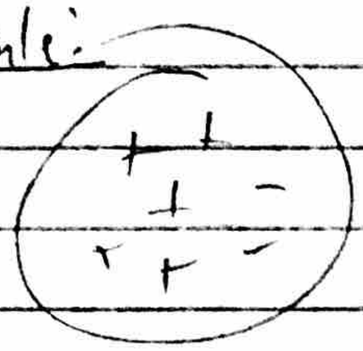
$G = 0.475$

Action has lowest gini so it is our first rule.

Tree so far:

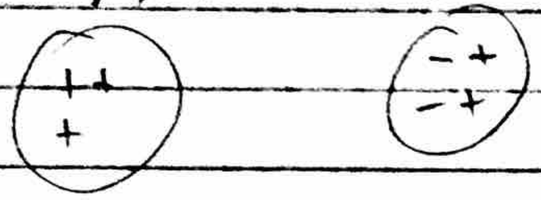


Try each remaining rule:



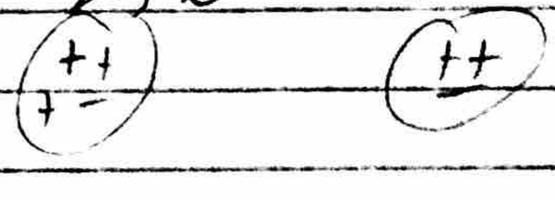
Budget > $100m?

yes ↙ ↘ no



Jennifer Lawrence?

yes ↙ ↘ no



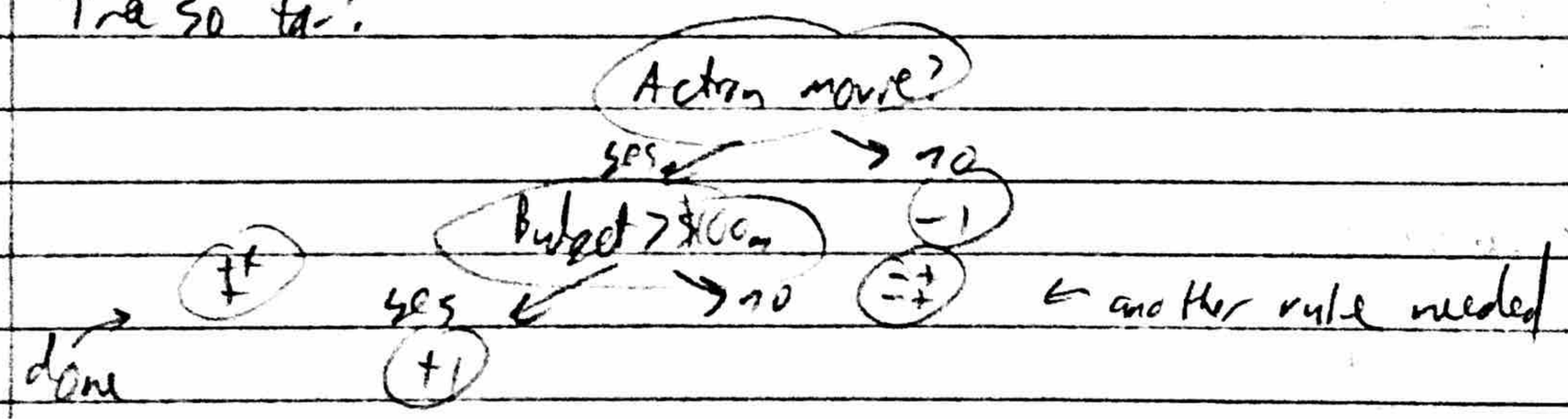
Which is better

G = 0.29

G = 0.41

Budget has lowest grn so it's the best rule.

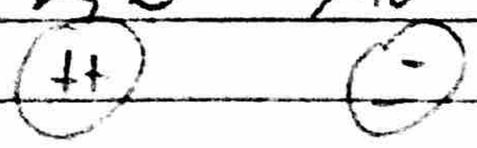
Tree so far:



Try each remaining rule:

Jennifer Lawrence?

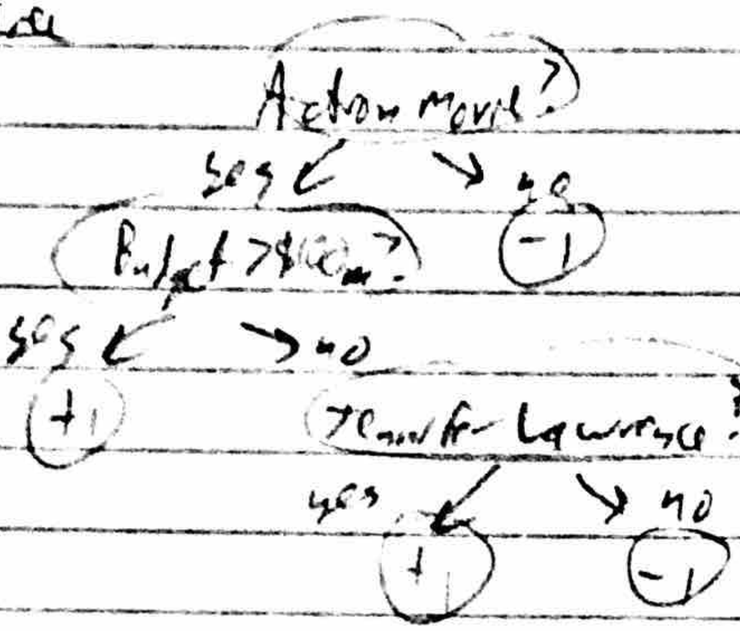
yes ↙ ↘ no



G = 0

Note: We won't always get a perfect split like this at the end.
For this particular data set our decision tree will get
100% accuracy, but that's not always the case.

Complete decision tree



Decision Forest

A single decision tree may be biased by a majority of the data points and will not be able to accurately classify the minority.

Idea: build multiple decision trees using different subsets of the data

The prediction of the decision forest is the (weighted or unweighted) prediction of the majority of the decision trees.

Random Forest

Problem: when selecting random subsets of the data, the overall majority tends to be the majority of the subset as well. This leads to similar decision trees.

(?) How can we increase the diversity of our decision trees?

Idea: Only try rules from a random subset of the available rules

We're still choosing the best of the available rules, but now we're enforcing diversity among the decision trees.

Random Forest Algorithm

repeat T times:
  select a random subset of the data
  while a node exists with gini > 0:
    select a node with gini > 0
    select a random subset of m variables
    for each variable:
      compute the gini index of the split
    select the variable with the best split and add it to the tree