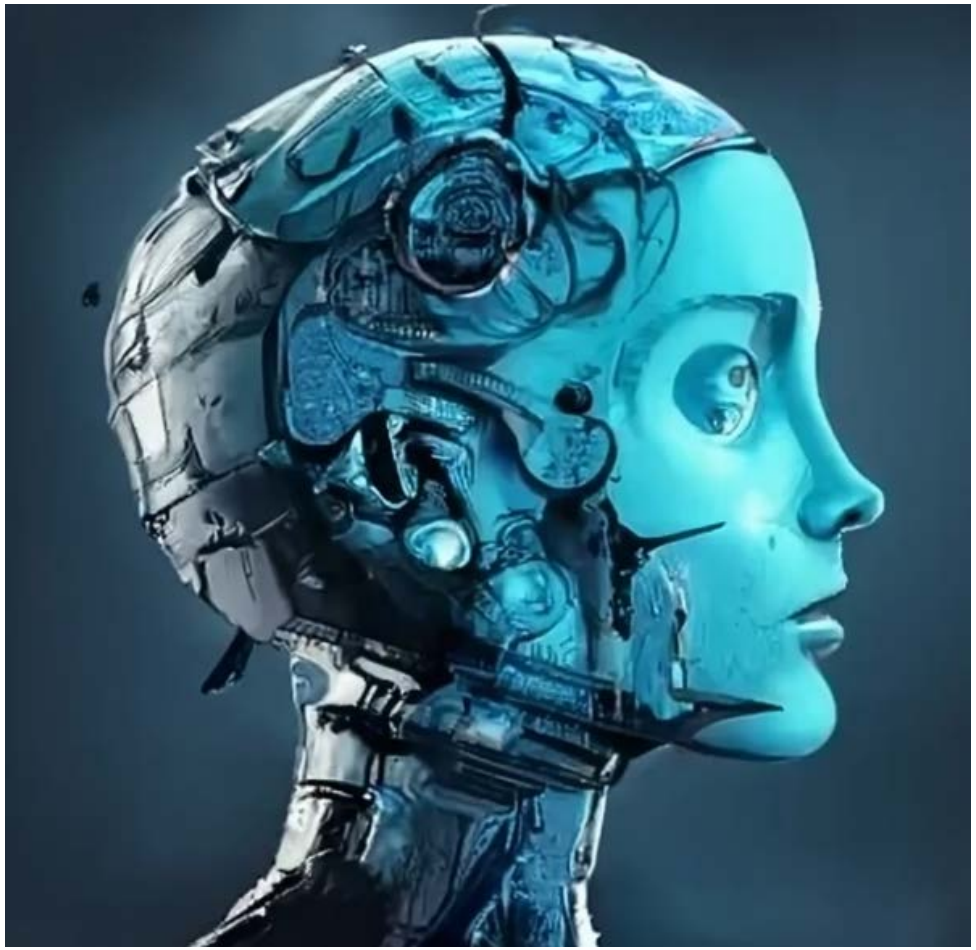
Machine Intelligence

Decision Trees

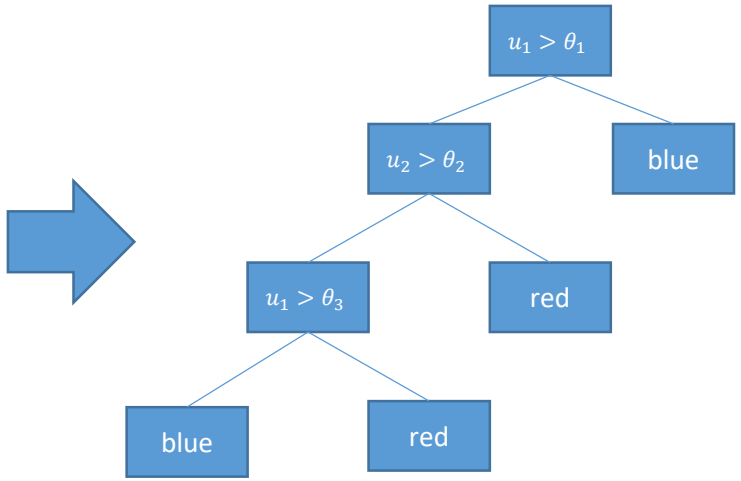
Dr. Cory Merkel

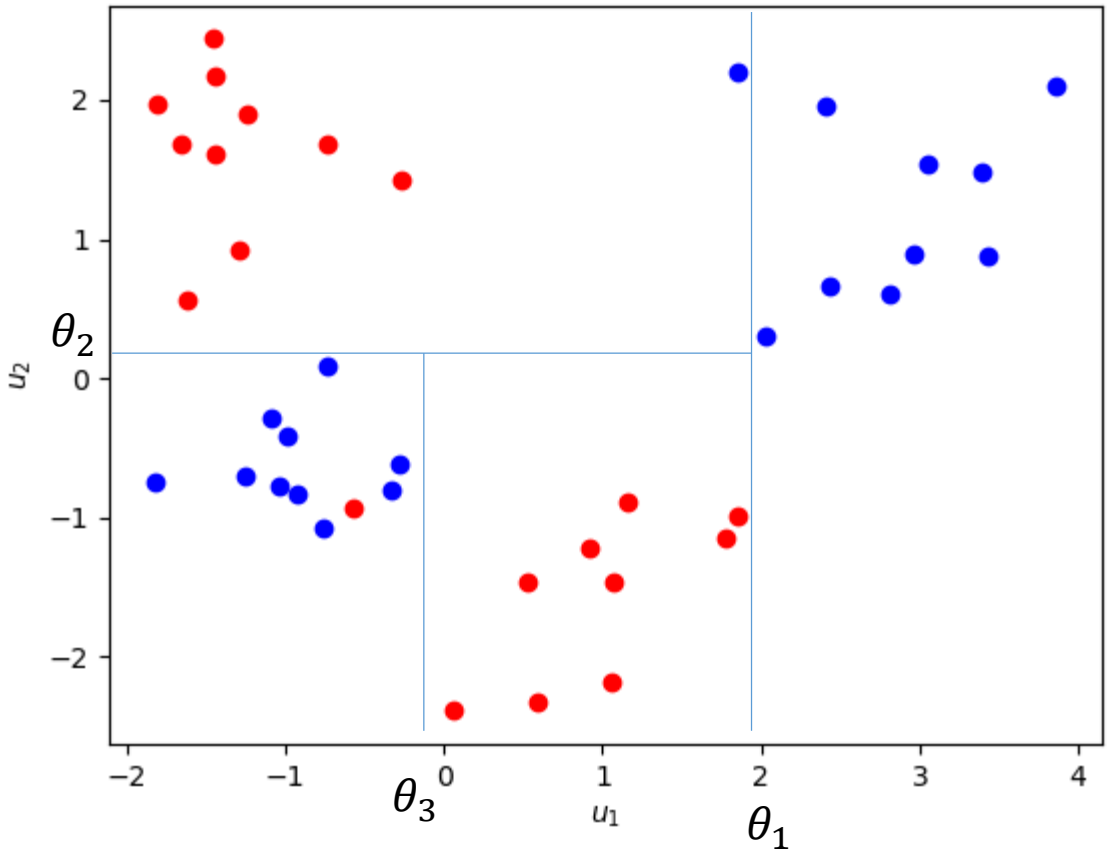


How Can We Recursively Define Decision Regions?

- For the data below, how could we break up the input space?
- One simple way is to force our decision regions to be hyperrectangles
  - Each decision boundary is orthogonal to one of the feature axes

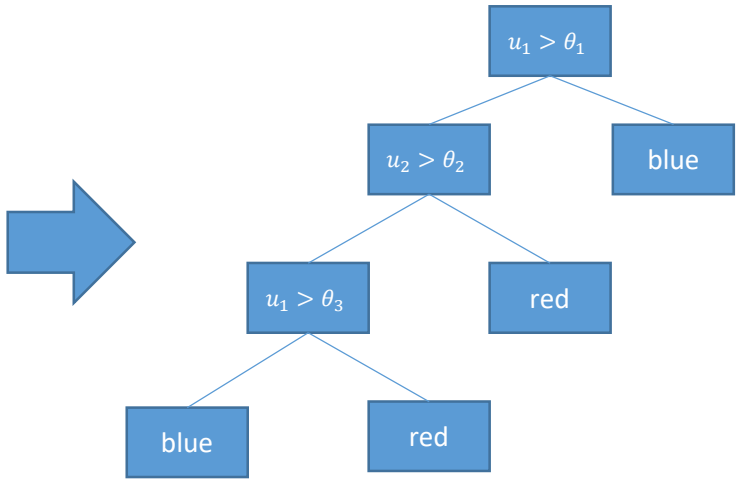
```
if u1 > θ1:
    blue class
else:
    if u2 > θ2:
        red class
    else:
        if u1 > θ3:
            red class
        else:
            blue class
```

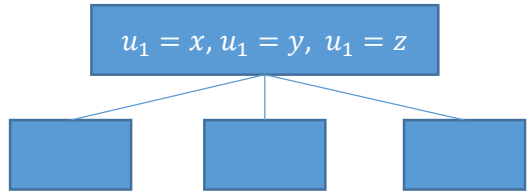




Decision Trees are Nested If-Else Statements

```
if u1 > θ1:
    blue class
else:
    if u2 > θ2:
        red class
    else:
        if u1 > θ3:
            red class
        else:
            blue class
```





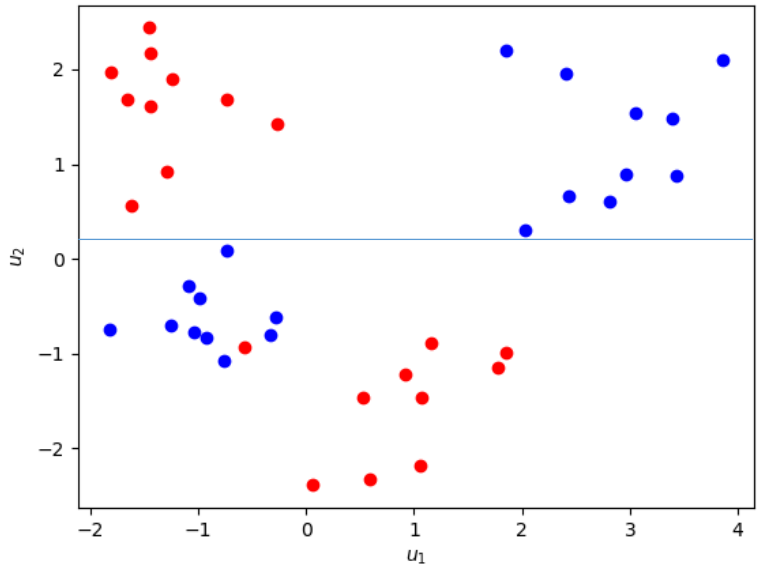
- Easy to interpret
- Can have multiple “splits” at each node
- Each level corresponds to a new decision boundary
- Each node corresponds to a region of the input space
- Each leaf corresponds to a classification

How Do We Build Decision Trees?

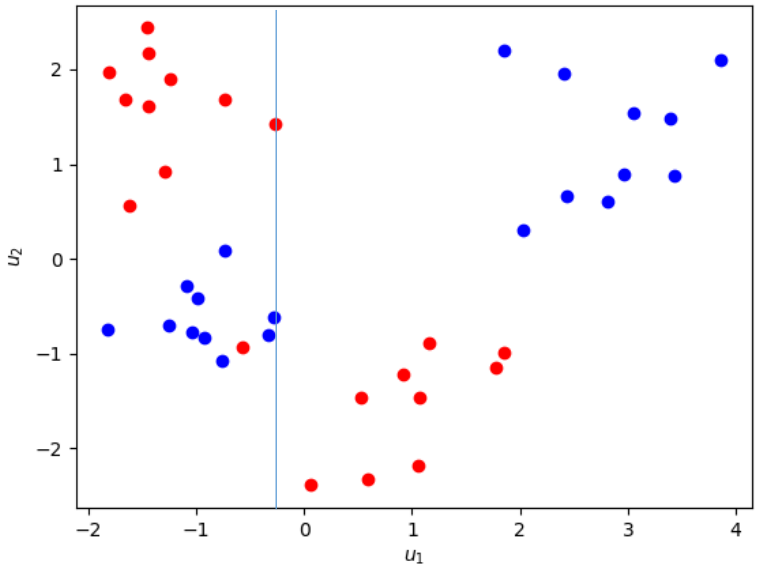
Feature + Set of conditions

- Start with entire input space
  - Choose a feature and one or more conditions to define decision boundaries
  - For binary trees, choose a feature and a threshold to split the input space in half
    - ❑Lots of ways we could do this:

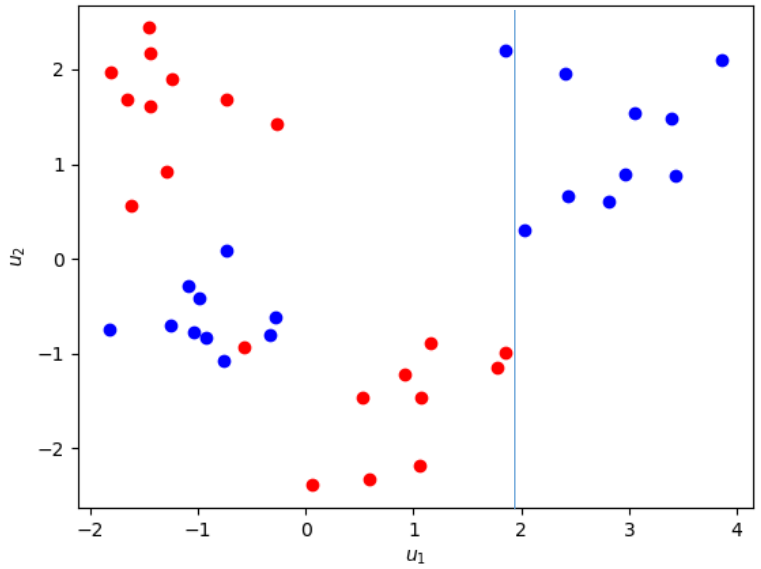
Split along u_2



Split along u_1



Split along u_1 with different threshold

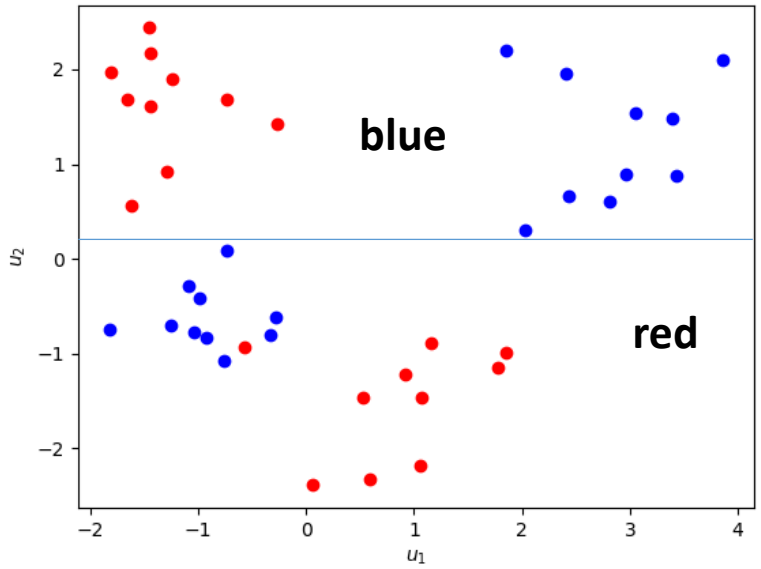


What is the Best Way to Split?

- When we decide to split our data, we should make sure that we have some better chance of correctly classifying the data after the split

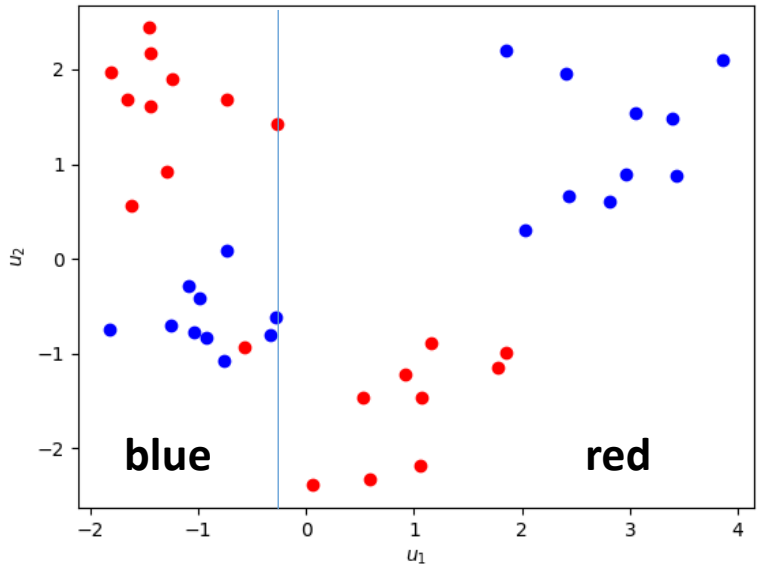
~ 50/50 chance of correct classification after split

Split along u_2



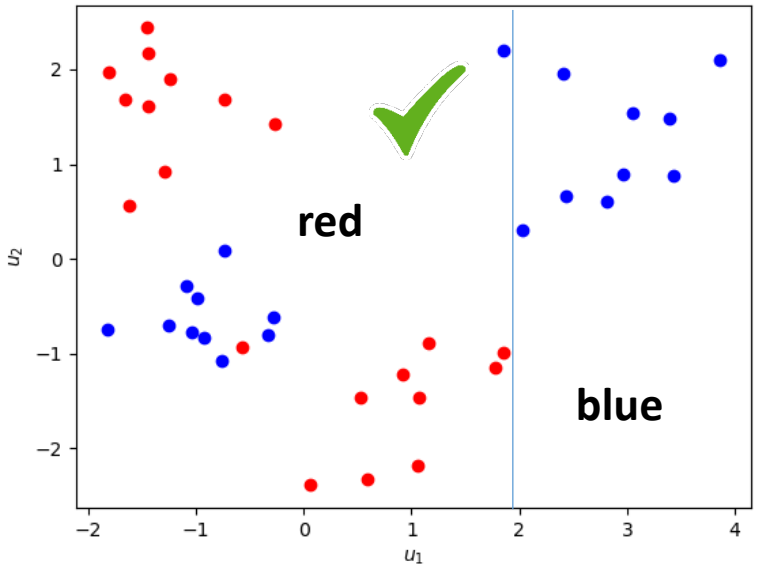
~ 50/50 chance of correct classification after split

Split along u_1



~ 75% accuracy after split

Split along u_1 with different threshold



What is the Best Way to Split?

- Another way of thinking about the best split is to consider how much new information the split gives us:
  - Same as saying, how much do we reduce our *uncertainty* of the data's class after the split

- Uncertainty of a random variable can be quantified using *entropy*

$$H \equiv -\sum_i p_i \log_2 p_i$$

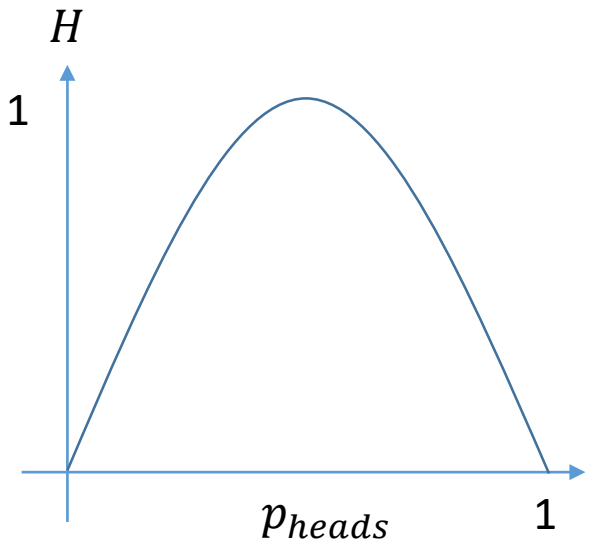
p_i is the probability of the i^{th} possible outcome of the random variable.

Entropy Example: Fair Coin Flip

- If we have a fair coin (equal probability of heads and tails), then we cannot confidently say whether it will come up heads or tails:
  - Entropy should be maximized in this case

$$H \equiv -\sum_i p_i \log_2 p_i = -p_{heads} \log_2 p_{heads} - p_{tails} \log_2 p_{tails}$$

- For $p_{heads} = (1 - p_{tails}) = 0.5, H = 1$
- As p_{heads} increases, we can more confidently say that the result will be heads, so entropy decreases.
  - For $p_{heads} = 1$, we are 100% certain of the outcome, so entropy should be 0



Information Gain

- We can look at the entropy before and after a split
  - If the entropy (uncertainty) is reduced after the split, then we have gained some information, and the split is good
  - If the entropy does not decrease, then we haven't gained any information

$$IG \equiv H_{before} - H_{after}$$

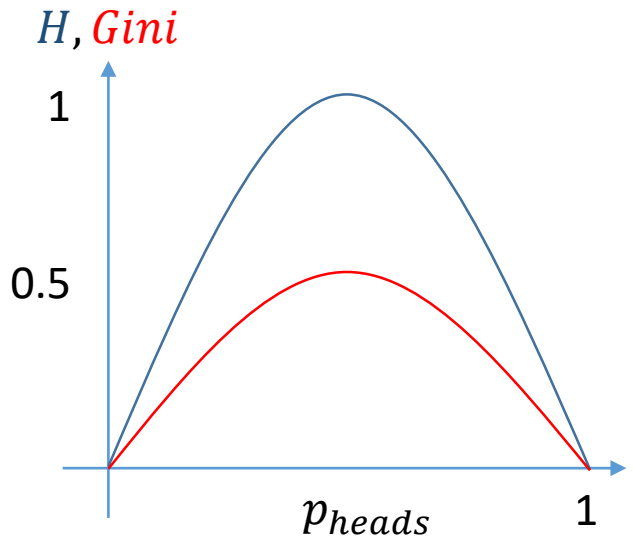
- IG is one of the main criterion used to determine how to make a split

Gini Index

- A similar metric to IG is the Gini index, which is defined as

$$Gini \equiv 1 - \sum_i p_i^2$$

- Note that Gini index can be though of as a measure of impurity
  - Want it to be small for good splits
  - Gini index is minimized when there is only one possible outcome



Weighting IG or Gini

- When we make a split, we need to know that the split is good on both sides
  - We can use a weighted average of IG or Gini on both sides of the split

Split along u_1 with different threshold

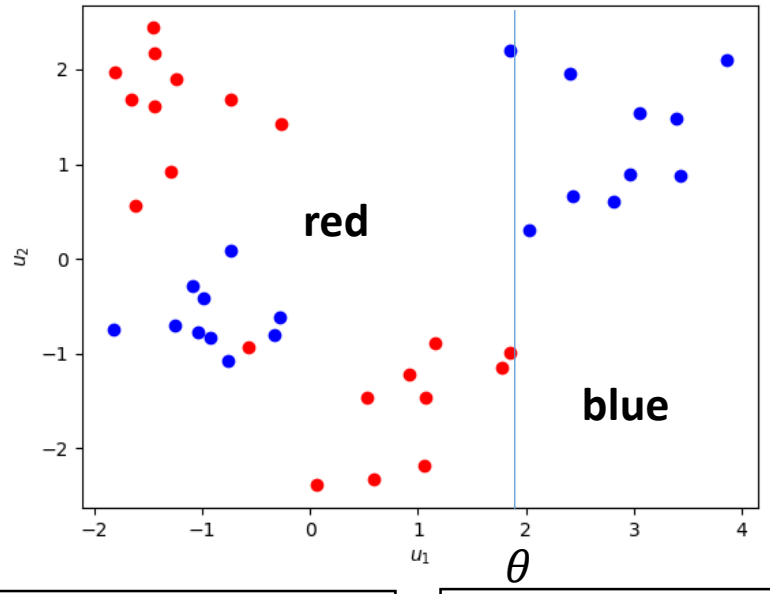
$$IG_{<\theta} = H_{before} - H_{after}$$

$$H_{before} = -\frac{20}{40}\log_2\frac{20}{40} - \frac{20}{40}\log_2\frac{20}{40} = 1$$

$$H_{after<\theta} = -\frac{11}{31}\log_2\frac{11}{31} - \frac{20}{31}\log_2\frac{20}{31} = 0.94$$

$$IG_{<\theta} = 0.06$$

$$Gini_{<\theta} = 1 - \left(\frac{11}{31}\right)^2 - \left(\frac{20}{31}\right)^2 = 0.46$$



$$IG_{>\theta} = H_{before} - H_{after}$$

$$H_{before} = -\frac{20}{40}\log_2\frac{20}{40} - \frac{20}{40}\log_2\frac{20}{40} = 1$$

$$H_{after>\theta} = -\frac{9}{9}\log_2\frac{9}{9} = 0$$

$$IG_{>\theta} = 1$$

$$Gini_{>\theta} = 1 - \left(\frac{9}{9}\right)^2 = 0$$

$$IG = p_{>\theta}IG_{>\theta} + p_{<\theta}IG_{<\theta} = \frac{9}{40}\times 1 + \frac{31}{40}\times 0.06 = 0.27$$

$$Gini = p_{>\theta}Gini_{>\theta} + p_{<\theta}Gini_{<\theta} = \frac{9}{40}\times 0 + \frac{31}{40}\times 0.46 = 0.36$$

ID3 Algorithm (Iterative Dichotomiser 3)

- Now we can measure IG or $Gini$ index to look at for all possible values of all features
  - Best feature/value combo is used for split
  - Iterate for a maximum tree depth or specified accuracy level

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

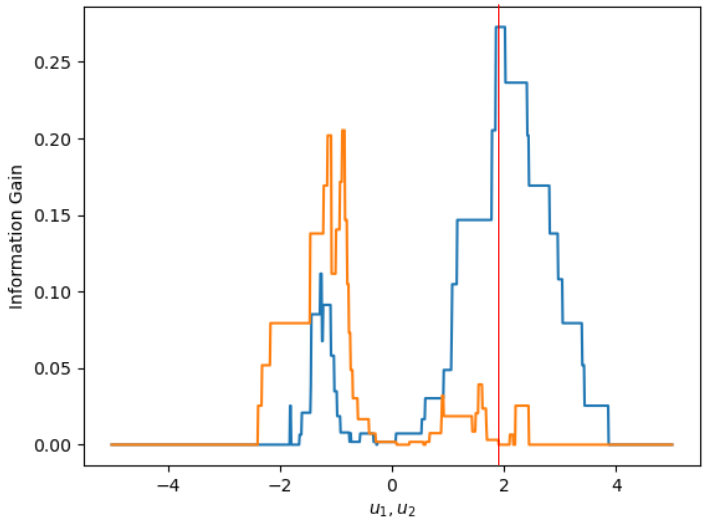
- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

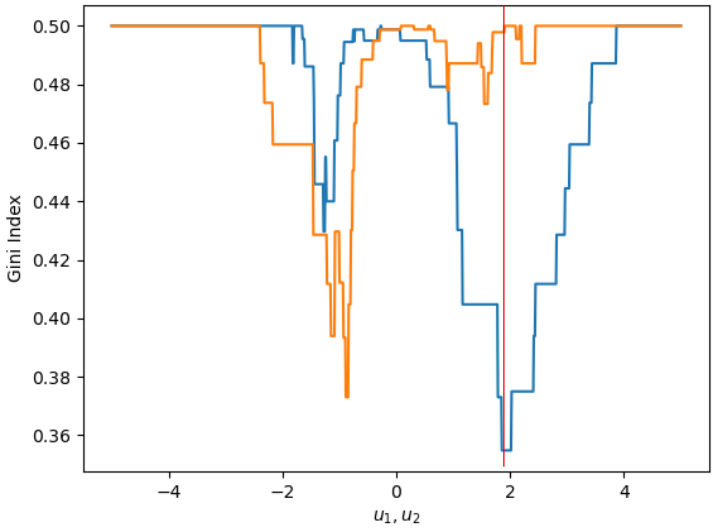
* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

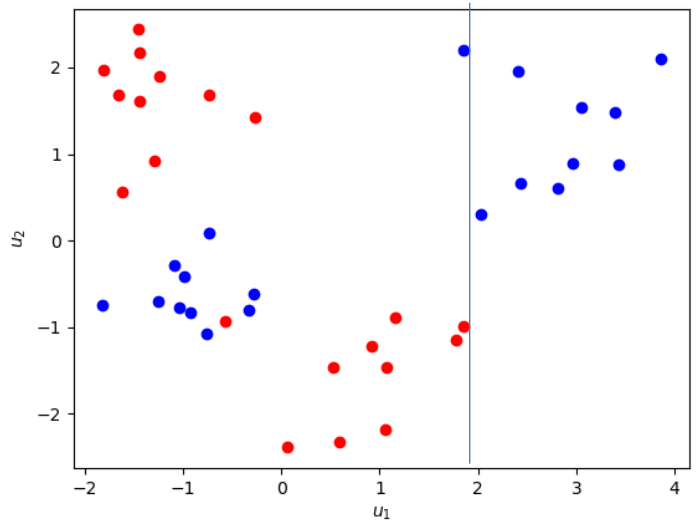
Tom Mitchell, Machine Learning

Example

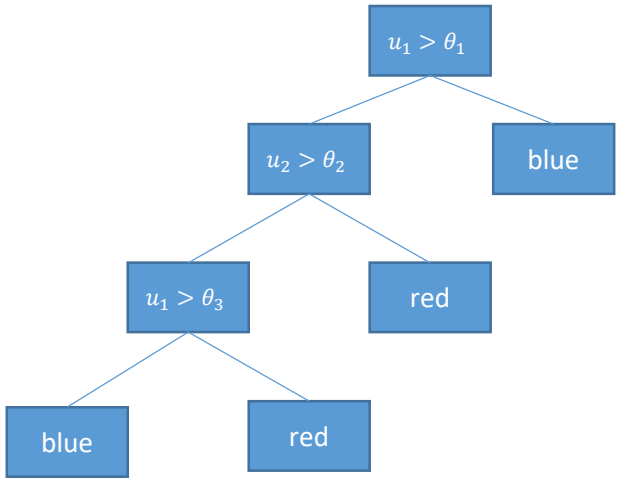
- Start with all of the data and measure IG or Gini index for all values of both u_1 and u_2
- Look for maximum of (weighted) IG or minimum of Gini index
  - In this example, both correspond to u_1 at the same threshold
- Make split at the corresponding threshold value
- Repeat for each side of the split…

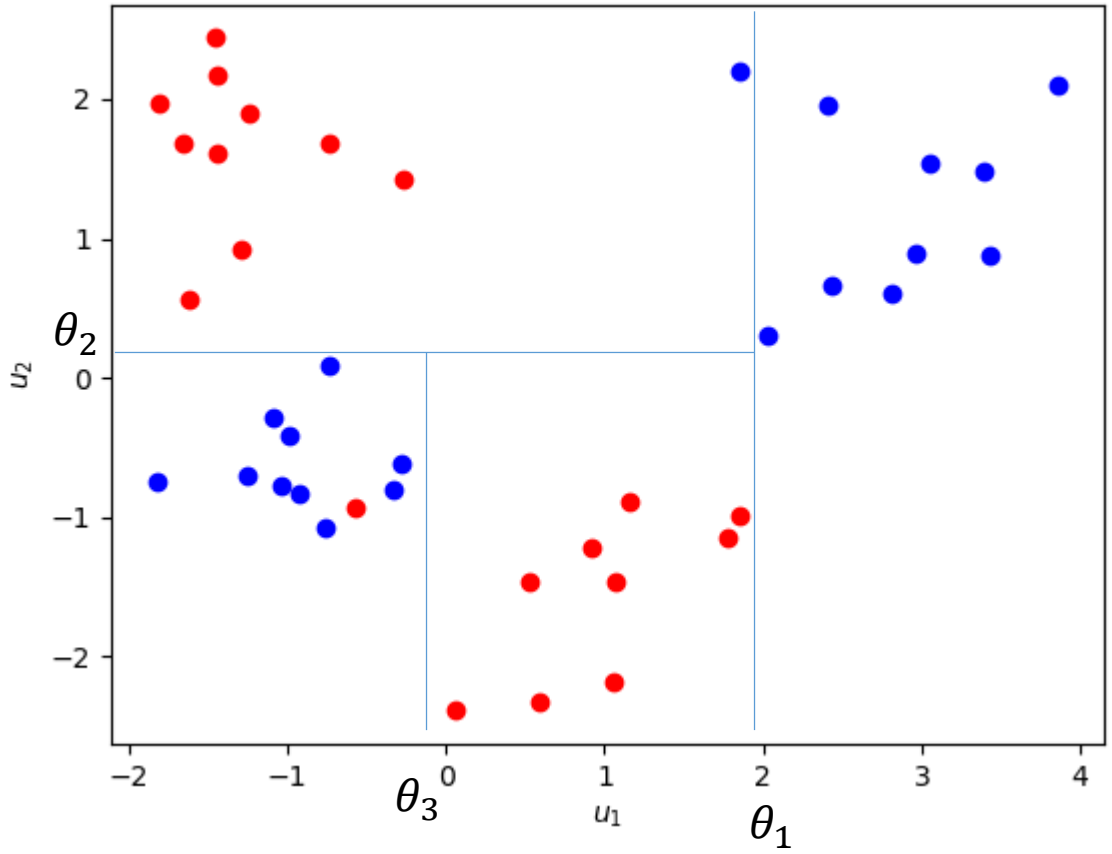






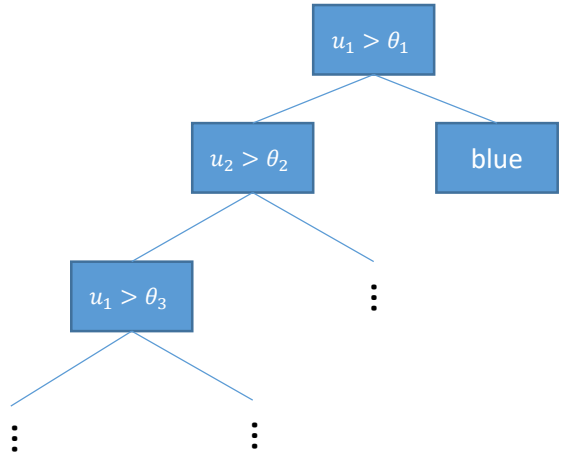
Final Tree for Depth of 4

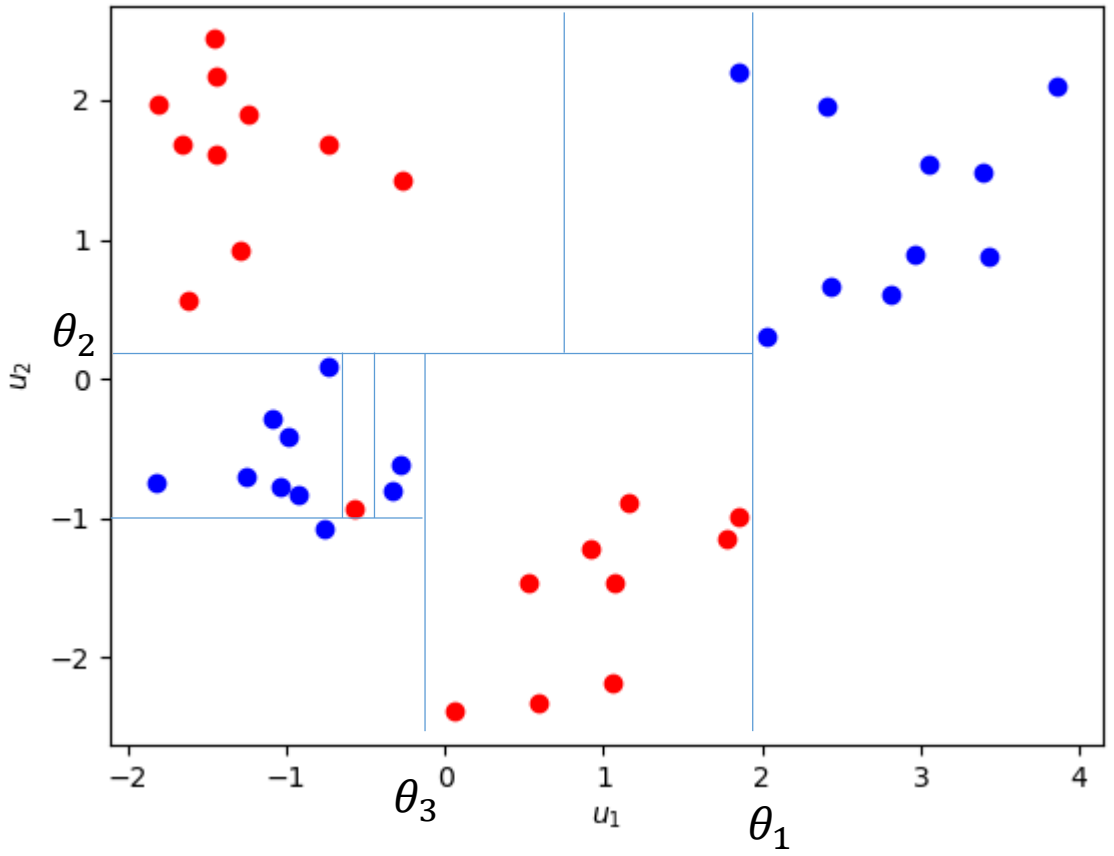




If We Kept Going…

- We could keep going until everything is classified correctly
  - Deeper tree
    - ❑Possible overfitting!





Another Example

- Decision trees are also very good for the case where features are discrete:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Tom Mitchell, Machine Learning

Another Example

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Given our tennis data, where should we make our first split?

$$p(no) = \frac{5}{14}, \qquad p(yes) = \frac{9}{14}$$

$$H_{before} = -p(no)\log_2\big(p(\text{no})\big) - p(yes)\log_2(p(yes)) = 0.9403$$

Split on outlook:

$$H_{sunny} = -p(no|sunny)\log_2\big(p(no|sunny)\big) - p(yes|sunny)\log_2\big(p(yes|sunny)\big)$$

$$= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.9710$$

Similarly:

$$H_{rain} = 0.9710, \qquad H_{overcast} = 0$$

- Entropy after split is weighted sum of entropies for each possible value of the feature:

$$H_{after} = p(sunny)H_{sunny} + p(rain)H_{rain} + p(overcast)H_{overcast}$$
$$= \left(\frac{5}{14}\right)(0.9710) + \left(\frac{5}{14}\right)(0.9710) + \left(\frac{4}{14}\right)(0) = 0.6936$$

- Information gain: $IG_{outlook} = H_{before} - H_{after} = 0.2467$

Another Example

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- We can also calculate the IG for Temperature, Humidity, and Wind, and we find

$$IG_{outlook} = 0.2467$$
$$IG_{humidity} = 0.151$$
$$IG_{wind} = 0.048$$
$$IG_{temp} = 0.029$$

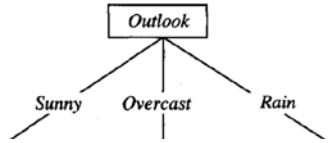
- Therefore, we make our first split based on outlook!
  - Would we get the same result using Gini Index?

Another Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Now, we look at each split based on outlook and calculate the best split within each one
- For the sunny split:

$$H_{before} = H_{sunny} = 0.9710$$



- Now, where should be split? Let's try temperature:

$$H_{hot} = -p(no|hot, sunny)\log_2 p(no|hot, sunny) - p(yes|hot, sunny)\log_2 p(yes|hot, sunny)$$
$$= -1\log_2 1 - 0\log_2 0 = 0$$

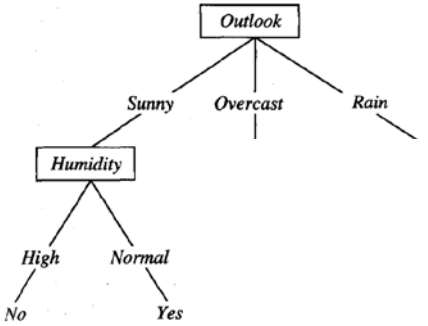
$$H_{mild} = 1$$
$$H_{cool} = 0$$
$$H_{after} = \left(\frac{2}{5}\right)(0) + \left(\frac{2}{5}\right)(1) + \left(\frac{1}{5}\right)(0) = 0.4$$
$$IG_{temp} = H_{before} - H_{after} = 0.5710$$

Another Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Similarly, we find:

$$IG_{temp} = 0.5710$$
$$IG_{humidity} = 0.971$$
$$IG_{wind} = 0.02$$

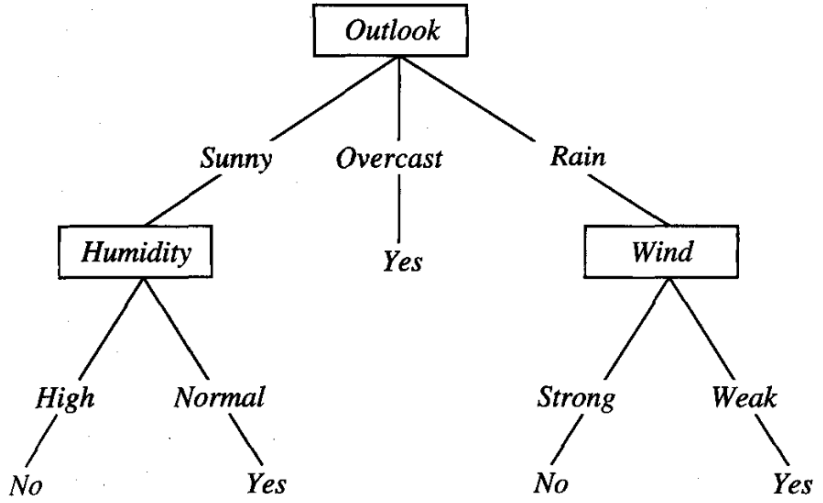


- So, for the sunny split, we make the next split on humidity
  - Note that after the humidity split, we cannot gain any more information
    - ❑ Before the split, we had $H_{sunny} = 0.971$, and after the split we gained 0.971 bits of information
    - ❑ We have all information to make a correct classification after the humidity split!

Another Example

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Keep going for other outlook possibilities, and eventually we find the final tree:



- Note that for decision trees with discrete features, we typically only use a feature once
- Also, notice that temperature turned out to be irrelevant to our decision!