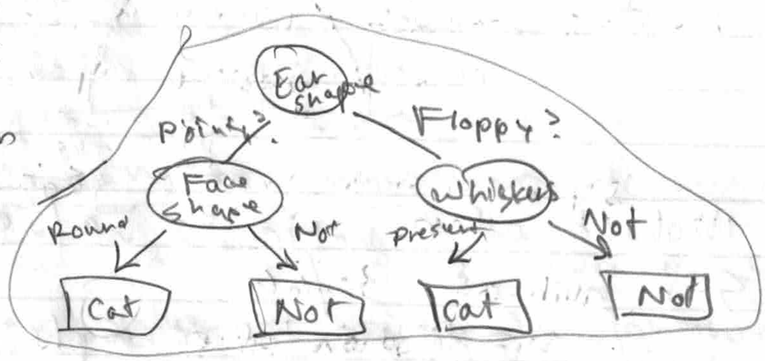
DECISION TREES

Super practical, great results, but not much academic interest!

Example:

- Build from data set
- Can build various trees from the data!



Learning a decision tree

1. Pick root node split feature
2. For each node in layer N+1, choose split feature
3. Recur until reaching "pure" nodes — fully segmented classes

Key decisions:

→ Choosing split features — want to (min) max purity
→ When to stop splitting?
- node is 100% one class
- reached max depth
- Purity score improvements don't meet threshold
- # of examples in node is below threshold

☆ Still need to avoid overfitting!

$\log_e x = y \Leftrightarrow e^y = x \Leftrightarrow 2^{ay} = x \Leftrightarrow 2^{ay} = x \Leftrightarrow \log_2 x = \log_2 \ldots y$

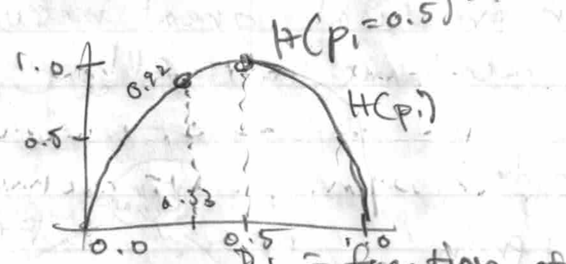
$2^a = e \Leftrightarrow \log_2 e = a$

$\Leftrightarrow \boxed{y = \frac{\log_2 x}{\log_2 e}}$

So if $\log_e x = y$
then $\log_2 x = y \cdot \log_2 e$

Measuring purity — Entropy

Entropy
H



p_1 = fraction of examples that are class 1

Highest when ratio is 50/50!
Still high at 33/66!

p_1 = fraction of examples w/ label 1

Define $p_0 = 1 - p_1$

Define:

$$H(p_1) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$
$$= \boxed{-p_1 \log_2(p_1) - (1-p_1)\log_2(1-p_1)}$$

$H(0) = 0$

Choosing a Split — Information Gain = Reduction of Entropy

For each possible split feature, compute H with that feature — but weighted by # examples — for both branches it generates. Then compare to "no split" to get Information Gain i.e. reduction in entropy.

Information Gain $= \boxed{H(p_1^{root}) - (w^{left} H(p_1^{left}) + w^{right} H(p_1^{right}))}$

where w^{left}, w^{right} are the fraction of the examples falling in the left/right nodes.

Putting it together

Repeatedly until stopping criteria met (max depth, info gain below t_1, examples below t_2)
- for each feature, compute info gain (IG)
- choose the one that maximizes IG
- recursion on L/R subtrees

Feature Encoding for non-binary features

- Categorical (non-binary) — use one-hot encoding
- Continuous values — pick a decision threshold by maximizing info gain

REGRESSION TREES

- Pick splits to reduce (weighted sum of) variance in subsets.
  maximize (root node var) − (weighted sum)
- Final prediction = mean of leaf node examples

TREE ENSEMBLES

→ Decision trees are highly sensitive to changes in training data so we train a tree ensemble and have them vote on a prediction. This increases robustness.

How? Random Forest algorithm.
- Use sampling with replacement to build data sets
- For each, build a decision tree (Bagged Decision Tree)
- For each node, only allow splits from $k < n$ random features (BGT → Random Forest). $k = \sqrt{n}$

XGBoost / Boosted Decision Trees

Most common decision tree algorithm.

Modify Random Forest alg by, for each new random tree, increase prob. of selecting samples that are misclassified by previous trees.

→ Emphasizes samples we're getting wrong

★ eXtreme Gradient Boost

- built in regularization
- good default hyperparams
- highly competitive
- fast to train!

(Also: might be more interpretable?)

When to use decision trees?

Tabular / structured data → XGBoost
(regression and classification)
(categorical and continuous)

Unstructured data (images, audio, text) → NNs
(works well on structured & semi-structured too)
Pros: transfer learning, can combine many models
Cons: slower