10-701 INTRODUCTION TO MACHINE LEARNING (PHD)
LECTURE 1: INTRO TO ML AND PERCEPTRON

LEILA WEHBE
CARNEGIE MELLON UNIVERSITY
MACHINE LEARNING DEPARTMENT

Lecture based on material from Tom Mitchell's lecture 2 (http://www.cs.cmu.edu/~tom/10701-S20/Intro-DTreesAndOverfitting-1-13-2020.pdf), Nina Balcan's lecture 2, and on on Kilian Weinberger's lecture 17 (https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote17.html)

LECTURE OUTCOMES

- What is a decision tree
- How to use information gain as a heuristic to building a short tree
- Notion of overfitting

LINKS (USE THE VERSION YOU NEED)

- Notebook (https://github.com/lwehbe/10701/blob/main/Lecture_02_decision_trees.ipynb)
- PDF slides (https://github.com/lwehbe/10701/raw/main/Lecture_02_decision_trees.pdf)

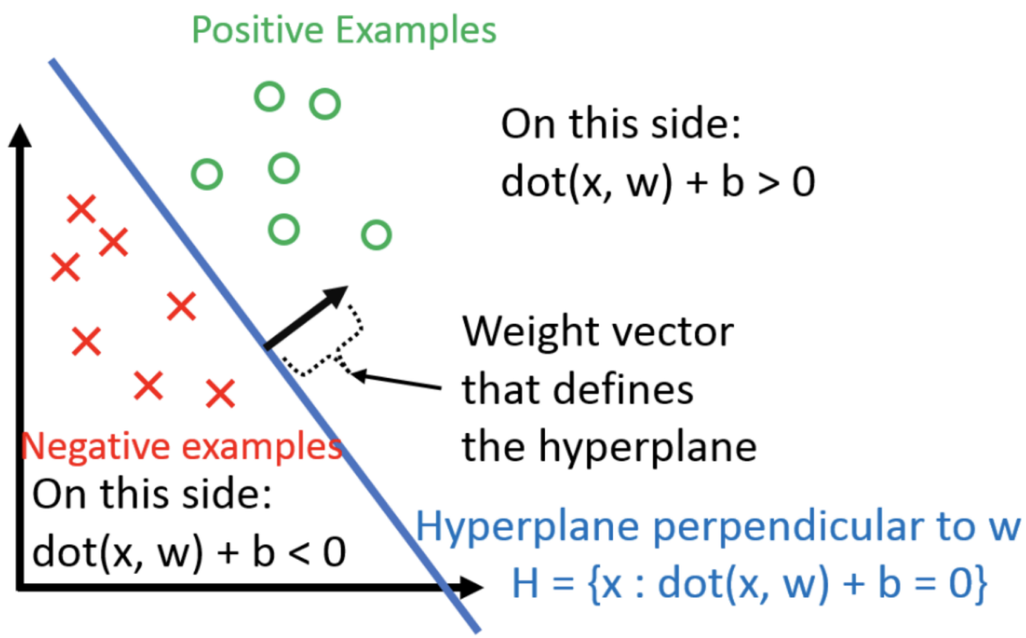
FIRST WE WILL FINISH THE PROOF FROM LAST LECTURE

THE PERCEPTRON

- Assume data is binary
- Assume data is linearly separable:
    - there exist a hyperplane that perfectly divides the two classes

$$\exists \mathbf{w}, b \quad \text{s.t.} \quad \forall (\mathbf{x}_i, y_i) \in D,$$
$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0$$

$$\exists \mathbf{w}, b \quad \text{s.t.} \quad \forall (\mathbf{x}_i, y_i) \in D,$$
$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) > 0$$



source (https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html)

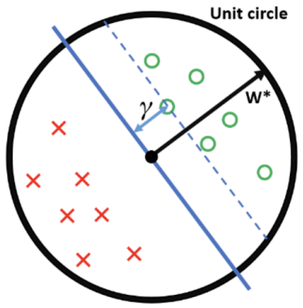
CONVERGENCE OF THE PERCEPTRON ALGORITHM

The perceptron algorithm converges in $\frac{1}{\gamma^2}$ updates if the data is linearly separable.

γ is the margin of the problem instance (defined on next slide).

NOTION OF MARGIN

- Assume there exists $\mathbf{w}^*$ such that $\forall(\mathbf{x_i}, y_i) \in D, \quad y_i(\mathbf{x_i}^\top \mathbf{w}^*) > 0$
- Also assume we rescale $\mathbf{w}^*$ and $\mathbf{x_i}$s such that:
    - $||\mathbf{w}^*|| = 1$ and $||\mathbf{x_i}|| \leq 1 \ \ \forall \mathbf{x_i}$ (how?)
- The margin γ of the hyperplane $\mathbf{w}^*$ is the minimum distance between one of the points and the hyperplane:
    - $\gamma = \min_{(\mathbf{x_i}, y_i) \in D} |\mathbf{x_i}^\top \mathbf{w}^*|$ (since $\mathbf{w}^*$ is unit norm)



source (https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html)

THEOREM

- Given:
    - All $\mathbf{x}_i$ s are within the unit sphere
    - There exists a separating hyperplane $\mathbf{w}^*$, with $||\mathbf{w}^*|| = 1$
    - γ is the margin of hyperplane $\mathbf{w}^*$
- If all of the above holds, then the Perceptron algorithm makes at most $\frac{1}{\gamma^2}$ mistakes.

- Would we want a large margin or a small margin?
- What types of datasets will converge quickly?

PROOF

source (https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote03.html)

Keeping what we defined above, consider the effect of an update ($\mathbf{w}$ becomes $\mathbf{w} + y\mathbf{x}$) on the two terms $\mathbf{w}^\top \mathbf{w}^*$ and $\mathbf{w}^\top \mathbf{w}$. We will use two facts:

- $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise we wouldn't make the update.
- $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

1. Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}^*$:

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + y(\mathbf{x}^\top \mathbf{w}^*) \geq \mathbf{w}^\top \mathbf{w}^* + \gamma$$

The inequality follows from the fact that, for $\mathbf{w}^*$, the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ (i.e. $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$).

This means that for each update, $\mathbf{w}^\top \mathbf{w}^*$ grows by **at least** γ.

2. Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}$:

$$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2(\mathbf{x}^\top \mathbf{x})}_{0 \leq \ \leq 1} \leq \mathbf{w}^\top \mathbf{w} + 1$$

The inequality follows from the fact that

- $2y(\mathbf{w}^\top \mathbf{x}) < 0$ as we had to make an update, meaning $\mathbf{x}$ was misclassified
- $0 \leq y^2(\mathbf{x}^\top \mathbf{x}) \leq 1$ as $y^2 = 1$ and all $\mathbf{x}^\top \mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$).

3. Now we know that after M updates the following two inequalities must hold:
   (1) $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$
   (2) $\mathbf{w}^\top \mathbf{w} \leq M$.
   We can then complete the proof:

$$
\begin{aligned}
M\gamma &\leq \mathbf{w}^\top \mathbf{w}^* && \text{By (1)} \\
&= \|\mathbf{w}\| \cos(\theta) && \text{by definition of inner-product, where } \theta \text{ is the angle between } \mathbf{w} \text{ and } \mathbf{w}^*. \\
&\leq ||\mathbf{w}|| && \text{by definition of cos, we must have } \cos(\theta) \leq 1. \\
&= \sqrt{\mathbf{w}^\top \mathbf{w}} && \text{by definition of } \|\mathbf{w}\| \\
&\leq \sqrt{M} && \text{By (2)} \\
\\
&\Rightarrow M\gamma \leq \sqrt{M} \\
&\Rightarrow M^2\gamma^2 \leq M \\
&\Rightarrow M \leq \frac{1}{\gamma^2} && \text{And hence, the number of updates } M \text{ is bounded from above by a constant.}
\end{aligned}
$$

- What happens if the data is not separable?
- Does the order matter?

FUNCTION APPROXIMATION

PROBLEM SETTING:

- Set of possible instances X
- Unknown target function $f : X \rightarrow Y$
- Set of candidate hypotheses $H = h|h : X \rightarrow Y$

INPUT:

- Training examples $\langle x^{(i)}, y^{(i)} \rangle$ of unknown target function f

OUTPUT:

- Hypothesis $h \in H$ that best approximates target function f

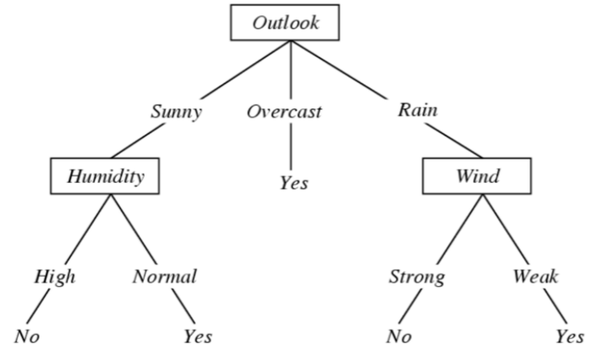
DECISION TREES

Learn concept PlayTennis (i.e., decide whether our friend will play tennis in a given day)

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

PLAY TENNIS?

- A Decision tree for f: (Outlook, Temperature, Humidity, Wind) → PlayTennis?



- Each internal node: test one discrete-valued attribute X_d
- Each branch from a node: selects one value for X_d
- Each leaf node: predict Y (or $P(Y|X \in \text{leaf})$)

Example: x=(Outlook=sunny, Temperature-Hot, Humidity=Normal,Wind=High), h(x)=Yes

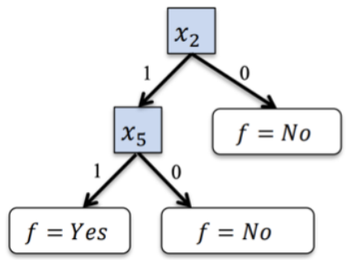
FUNCTION APPROXIMATION

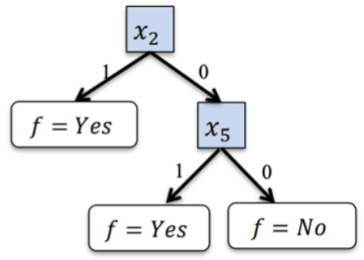
PROBLEM SETTING:

- Set of possible instances X
    - example: Outlook, Temperature, Humidity, Wind
- Unknown target function $f : X \rightarrow Y$
    - example: Y is binary (play or not play)
- Set of candidate hypotheses $H = h|h : X \rightarrow Y$
    - example: each h is a decision tree

DECISION TREE EXAMPLE

- Suppose $X = (X_1, X_2, \ldots X_n)$ where X_i are boolean-valued variables
- How would you represent $Y = X2 \wedge X5?$ $\quad$ $Y = X2 \vee X5?$





How would you represent $X_2X_5 \vee X_3X_4(\neg X1)?$

EXAMPLE WITH TRAINING DATA

X_1	X_2	X_3	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- The order we pick the attributes affects the size (and efficiency) of the tree.

HOW TO CHOOSE THE BEST HYPOTHESIS?

- How to automatically find a good hypothesis for training data?
    - A core algorithmic question.
- When do we generalize and do well on unseen data?
    - A learning theory question.
    - **Occam's razor**: use the simplest hypothesis consistent with data!
- *Occam's razor: Fewer short hypotheses than long ones*
    - *a short hypothesis that fits the data is less likely to be a statistical coincidence*
    - *highly probable that a sufficiently complex hypothesis will fit the data*
    - *(remember this is a heuristic. read here (https://machinelearningmastery.com/ensemble-learning-and-occams-razor/) for more discussion)*

HOW TO CHOOSE THE BEST HYPOTHESIS?

- How to automatically find a good hypothesis for training data?
    - A core algorithmic question.
- When do we generalize and do well on unseen data?
    - A learning theory question.
    - **Occam's razor**: use the simplest hypothesis consistent with data!
- Other core questions in machine learning:
    - How do we choose a hypothesis space?
        - Often we use prior knowledge to guide this choice
    - How to model applications as machine learning problems?
        - engineering challenge

HOW TO CHOOSE THE BEST DECISION TREE?

- How to pick the smallest tree?
    - NP-hard [Hyafil-Rivest'76]!
- Luckily, we have very nice practical heuristics and top down algorithms (e.g, ID3) that can achieve a good solution

TOP-DOWN INDUCTION OF DECISION TREES [EXAMPLES ID3, C4.5, QUINLAN]

- ID3: Natural greedy approach to growing a decision tree top-down (from the root to the leaves by repeatedly replacing an existing leaf with an internal node.).
- Algorithm main loop:
    - Pick “best” attribute A to split at a node based on training data.
    - Assign A to this node.
    - For each value of A create new descendent.
    - Split training examples among the descendents.
    - If training examples perfectly classified in new node, stop, else recurse on node.
- How to know which attribute is best?

WHICH ATTRIBUTE IS BEST?

- ID3 uses a statistical measure called **information gain** (how well a given attribute separates the training examples according to the target classification)
- Information Gain of variable A is the expected reduction in entropy of target variable Y for data sample $S4$, due to sorting on variable A

$$\text{Gain}(S, A) = H_S(Y) - H_S(Y|A)$$

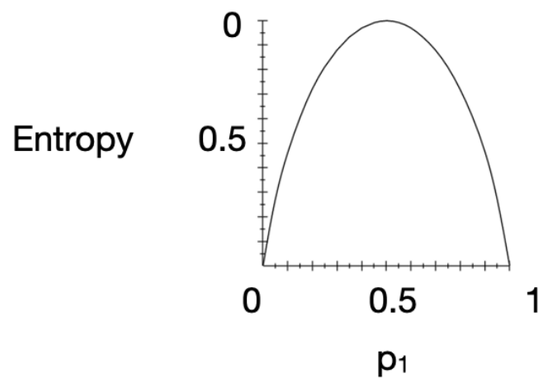
- Entropy (H_S) is information theoretic measure that characterizes the complexity of a labeled set S.

ENTROPY

- Sample Entropy of a Labeled Dataset
    - S is a sample of training examples
        - p_1 is the proportion of positive examples in S
        - p_2 the proportion of negative examples in S

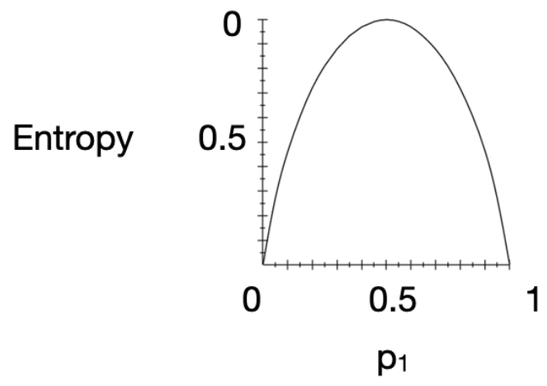
• Entropy measures the complexity of S:

$$H_S = -p_1 \log(p_1) - p_0 \log(p_0)$$



ENTROPY

$$H_S = -p_1 \log(p_1) - p_0 \log(p_0)$$



- E.g., if all negative, then entropy=0. If all positive, then entropy=0.
- If 50/50 positive and negative then entropy=1.
- If 14 examples with 9 positive and 5 negative, then entropy=.940

IF LABELS NOT BOOLEAN

$$H_S = -\sum_{k=1}^{c} p_k \log_2(p_k)$$

E.g., if c classes, all equally likely, then $p_k = \frac{1}{c}$ for all k and $H_S = -\log_2$

ANOTHER WAY TO UNDERSTAND MAXIMIZING INFORMATION GAIN

- We want to be the farthest possible from all classes being equally likely (let's call this uniform distribution $q_k = \frac{1}{c}$ for all k). This can be done by finding another distribution $(p_1, \ldots p_c)$ that maximizes the KL-Divergence (Kullback–Leibler divergence).
- The KL-Divergence is not a metric because it is not symmetric, i.e., $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$.)

$$KL(p||q) = \sum_{k=1} p_k log \frac{p_k}{q_k} \geq 0 \leftarrow KL\text{-Divergence}$$

$$= \sum_k p_k log(p_k) - p_k log(q_k) \text{ where } q_k = \frac{1}{c}$$

$$= \sum_k p_k log(p_k) + p_k log(c)$$

$$= \sum_k p_k log(p_k) + log(c) \sum_k p_k \text{ where } log(c) \leftarrow \text{constant}, \sum_k p_k = 1$$

$$\max_p KL(p||q) = \max_p \sum_k p_k log(p_k)$$

$$= \min_p - \sum_k p_k log(p_k)$$

$$= \min_p H(s) \leftarrow \text{Entropy}$$

$$H(S) = p^L H(S^L) + p^R H(S^R)$$

$$p^L = \frac{|S^L|}{|S|}, p^R = \frac{|S^R|}{|S|}$$

ENTROPY OF A SPLIT

- if binary:

$$H(S) = p_1 H(S_1) + p_0 H(S_0)$$
$$p_1 = \frac{|S_1|}{S}, \quad p_0 = \frac{|S_0|}{S}$$

- more generallly for c classes:

$$H(S) = \sum_{k=1}^{c} \frac{|S_k|}{S} H(S_k)$$

INFORMATION GAIN OF A SPLIT

- Information Gain of variable A is the expected reduction in entropy of target variable Y for data sample $S4$, due to sorting on variable A (with c values):

$$\text{Gain}(S, A) = H_S(Y) - H_S(Y|A)$$

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k=1}^{c} \frac{|S_k|}{S} \text{Entropy}(S_k)$$

- Gain(S,A) is the information provided about the target function, given the value of some other attribute A.

Learn concept PlayTennis (i.e., decide whether our friend will play tennis in a given day)

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

Which attribute is the best classifier?

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k=1}^{c} \frac{|S_k|}{S} \text{Entropy}(S_k)$$

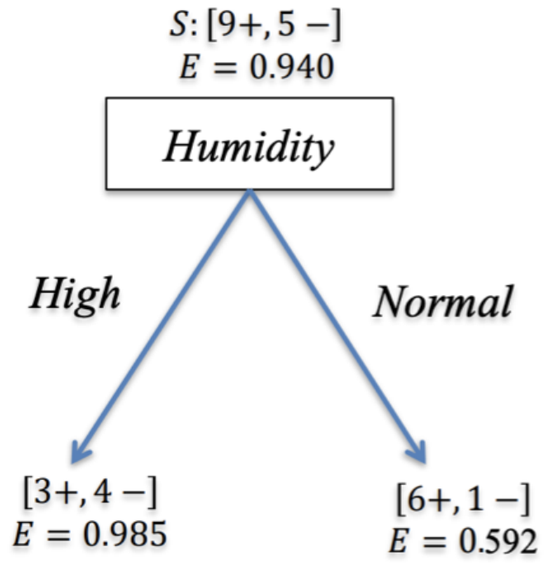
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

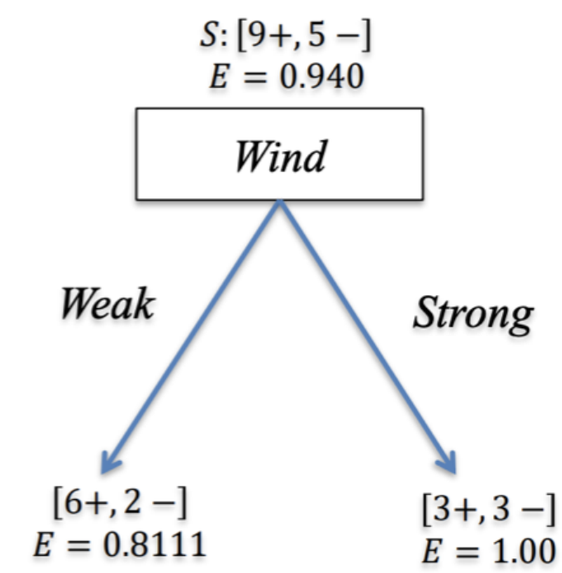
$$Entropy[9+, 5-] = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) = .940$$

Which attribute is the best classifier?

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k=1}^{c} \frac{|S_k|}{S} \text{Entropy}(S_k)$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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





$$Gain(S, Humidity) = .940 - \left(\frac{7}{14}\right).985 - \left(\frac{7}{14}\right).592 = .151$$

$$Gain(S, Wind) = .940 - \left(\frac{8}{14}\right).811 - \left(\frac{6}{14}\right)1.0 = .048$$

Which attribute is the best classifier?

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k=1}^{c} \frac{|S_k|}{S} \text{Entropy}(S_k)$$

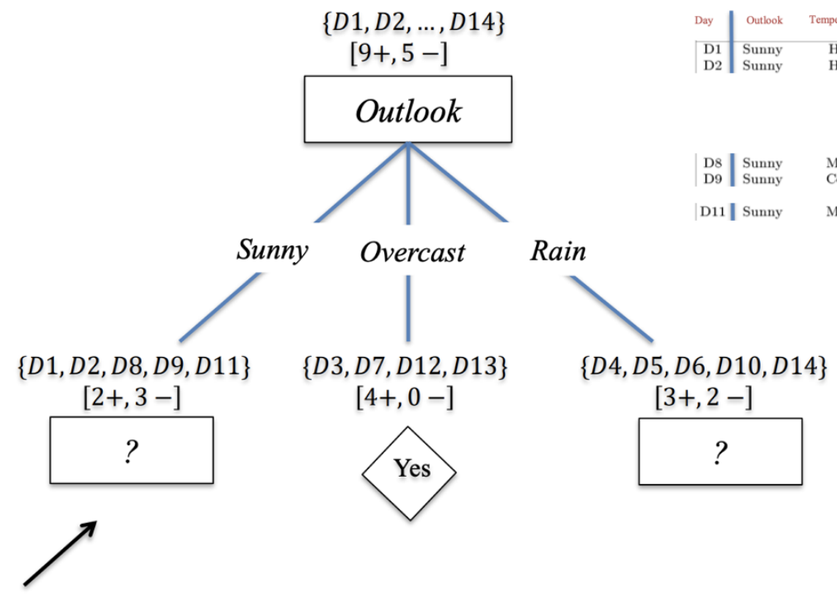
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

$Gain(S, Humidity) = .151$

$Gain(S, Wind) = .048$

$Gain(S, Outlook) = .246$

$Gain(S, Temperature) = .029$



Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Which attribute should be tested here?

$s_{sunny} = \{D1, D2, D8, D9, D11\}$

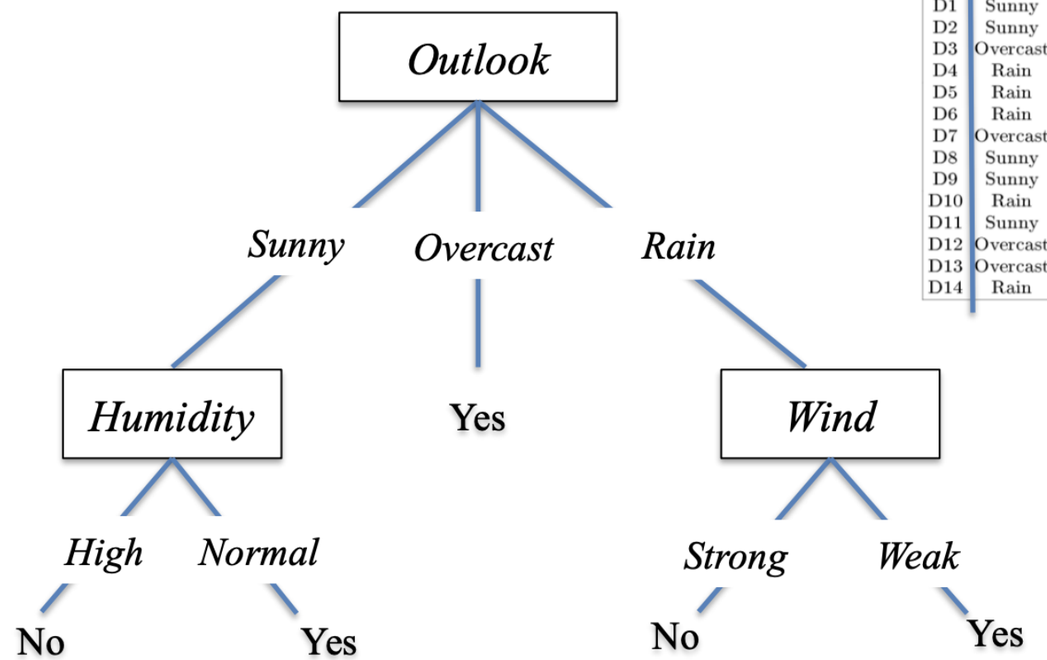
$Gain(s_{sunny}, Humidity) = .970 - \left(\frac{3}{5}\right)0.0 - \left(\frac{2}{5}\right)0.0 = .970$

$Gain(s_{sunny}, Temperature) = .970 - \left(\frac{2}{5}\right)0.0 - \left(\frac{2}{5}\right)1.0 - \left(\frac{1}{5}\right)0.0 = .570$

$Gain(s_{sunny}, Wind) = .970 - \left(\frac{2}{5}\right)1.0 - \left(\frac{3}{5}\right).918 = .019$

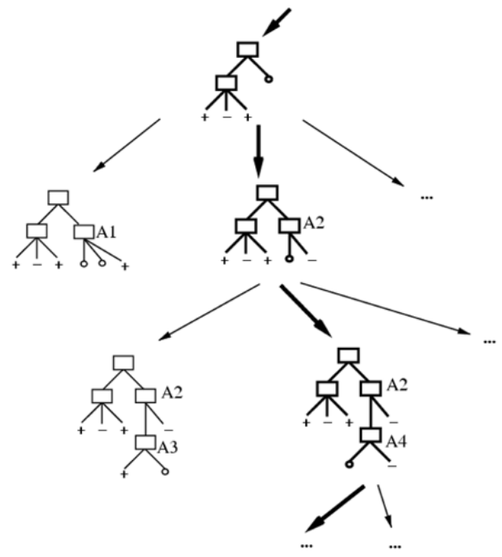
FINAL DECISION TREE

f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?



Day	Outlook	Temperature	Humidity	Wind	Play Tennis
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

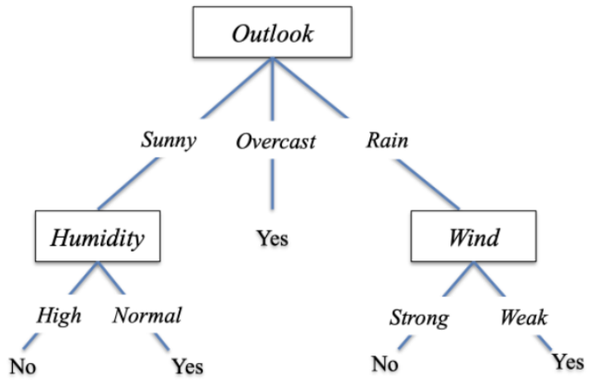
PROPERTIES OF ID3



- ID3 performs heuristic search through space of decision trees.
- It stops at smallest acceptable tree. (Occam's razor).
- Still a greedy approach, might not find the shortest tree.

ID3 MIGHT STILL OVERFIT!

- Overfitting could occur because of noisy data and because ID3 is not guaranteed to output a small hypothesis even if one exists.
- Consider adding a noisy example:
    - Sunny, Hot, Normal, Strong, PlayTennis = No
- The tree we learned would not be compatible with the training data



OVERFITTING

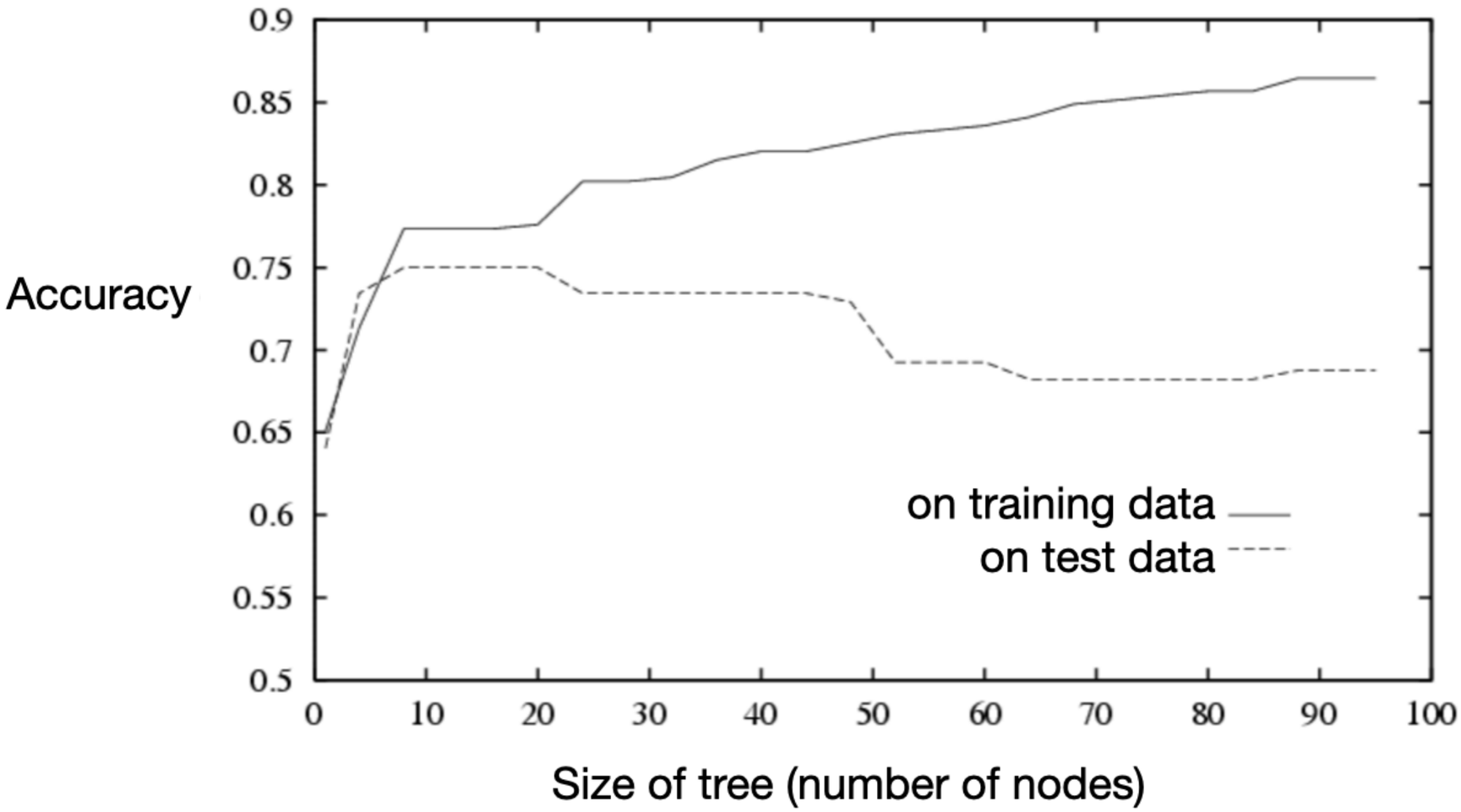
- Consider a hypothesis h and its
    - Error rate over training data: $errortrain(h)$
    - True error rate over all data: $errortrue(h)$

- We say h overfits the training data if $errortrue(h) > errortrain(h)$
    - We typically don't know $errortrue(h)$ but we can estimate $errortest(h)$ on a heldout set from the same distribution as the training data.

- Amount of overfitting = $errortrue(h)$
  $- errortrain(h)$

EXAMPLE OF OVERFITTING IN ID3

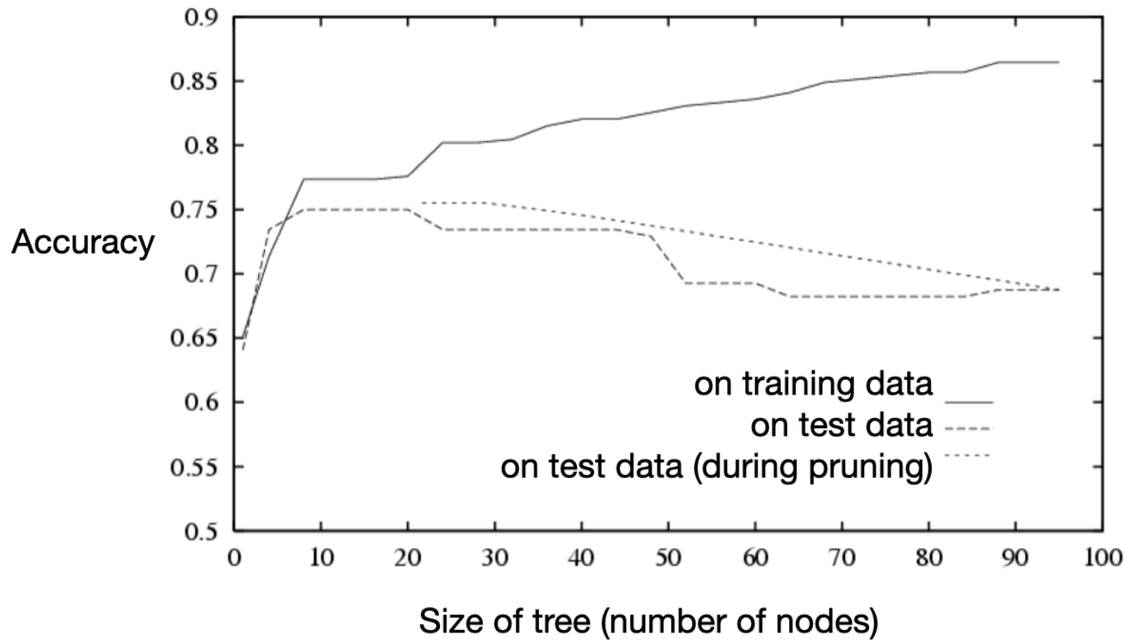
Task: learning which medical patients have a form of diabetes.



HOW CAN WE AVOID OVERFITTING?

- Stop growing when data split not statistically significant
- Grow full tree, then prune it
- example: Reduced Error Pruning
    - Split data into training set and validation set
    - Train a tree to classify training set as well as possible
    - Do until further pruning is harmful:
        1. For each internal tree node, consider making it a leaf node (pruning the tree below it)
        2. Greedily chose the above pruning step that best improves error over validation set
    - Produces smallest version of the most accurate pruned tree

EFFECT OF REDUCED ERROR PRUNING



NOTE: THE TEST SET SHOULD NEVER BE USED FOR TRAINING. A SEPARATE VALIDATION SET IS USED FOR MAKING DECISIONS ABOUT PRUNING.

WHAT IF MY ATTRIBUTES X ARE REAL VALUED?

- Use a decision stump: for each attribute, consider splitting above, below
    - e.g. (is Temperature ≥ 72)

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

ENSEMBLE LEARNING, BAGGING

- Using ensemble learning with trees makes them work very well in practice
    - instead of one tree, create a forest of trees and combine their prediction
- Bagging: resample the training dataset with replacement and average the trees
- Random forests: use subsets of the data (different features)
- Will see ensemble learning later in the couse