CSE 417T
Introduction to Machine Learning

Lecture 15

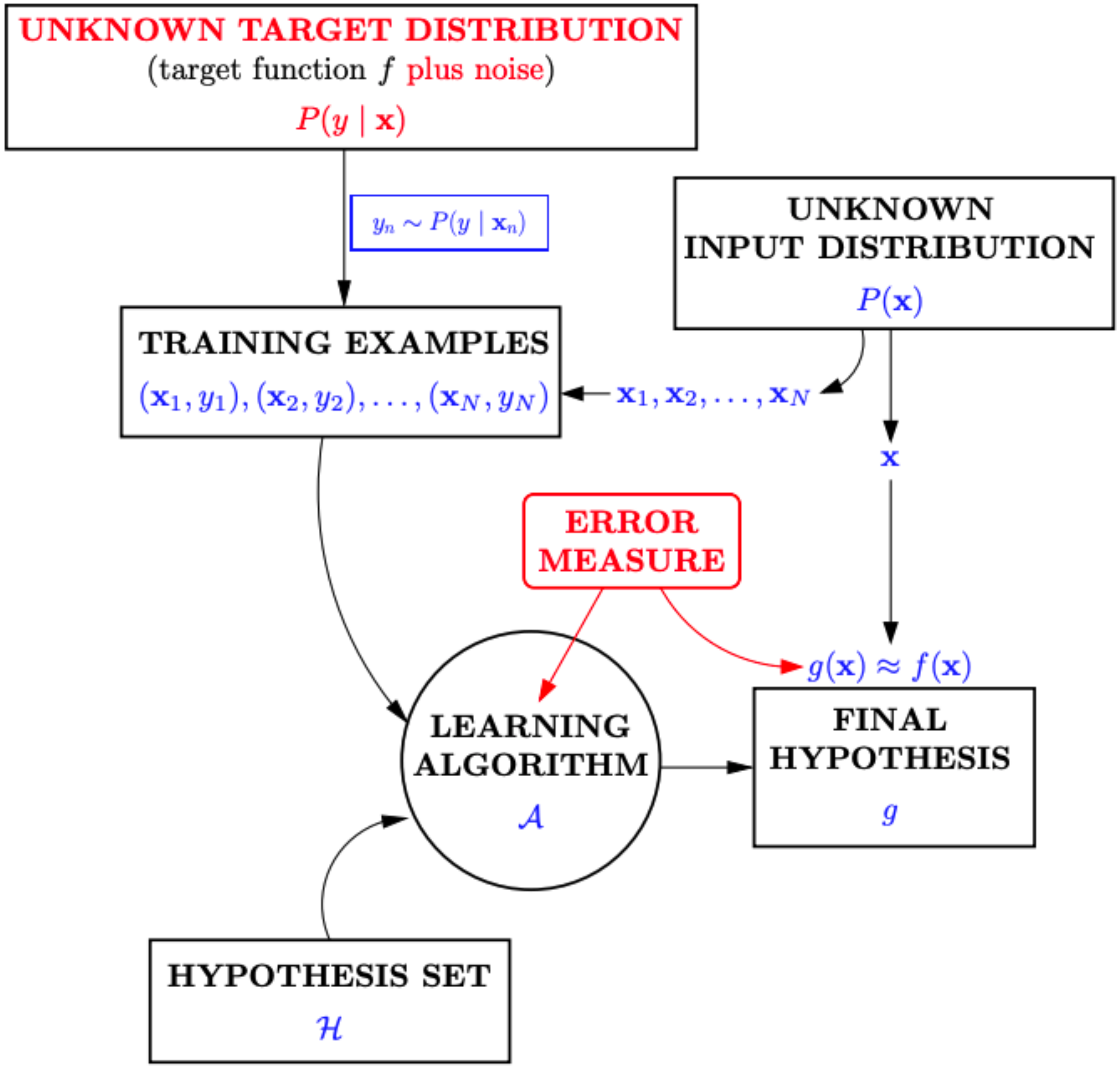
Instructor: Chien-Ju (CJ) Ho

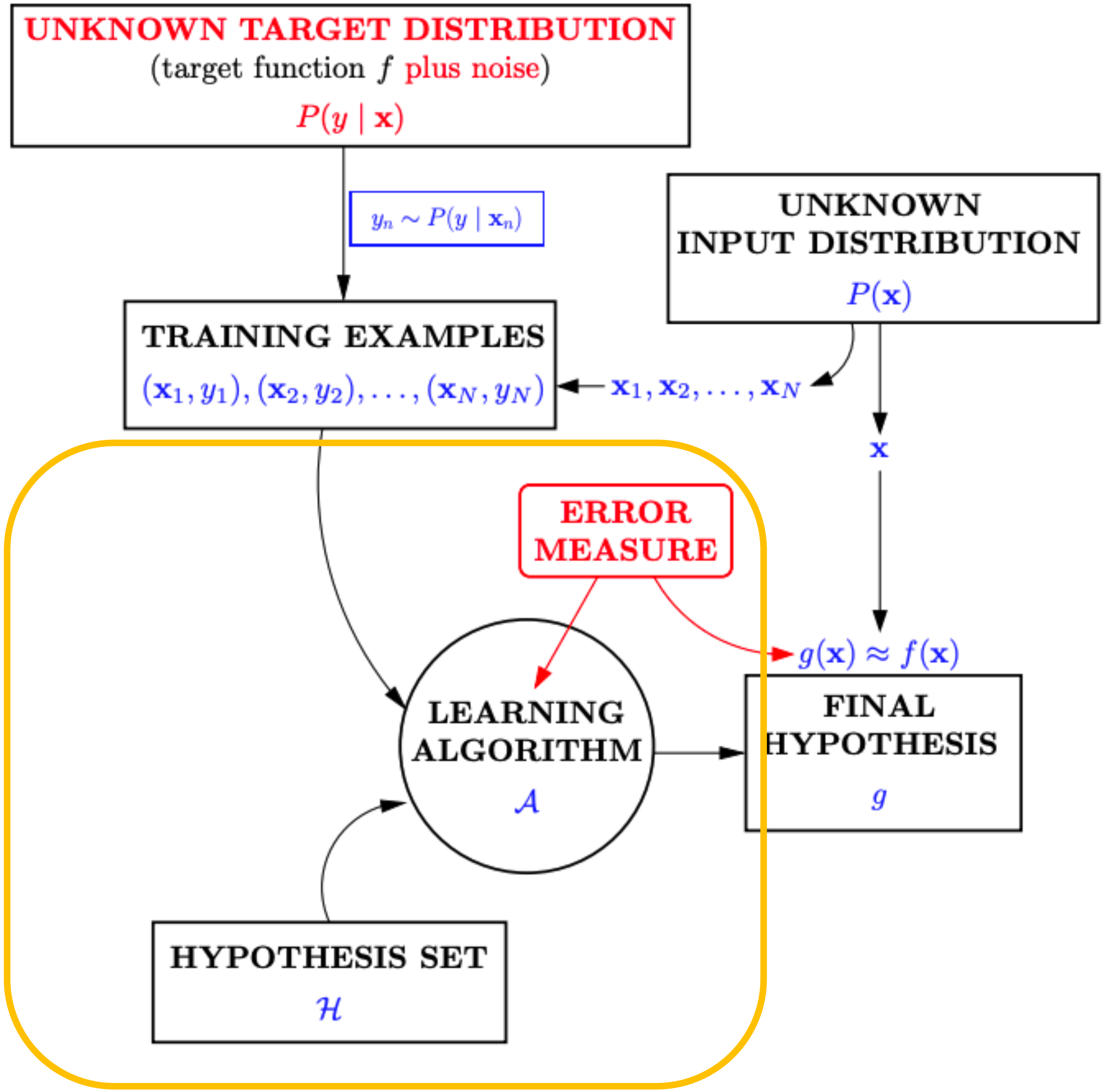
Logistics: Homework 3

- Homework 3 is posted on the course website
- Due on March 25 (Wednesday), 2020
  - Homework 4 will be announced before the due of homework 3

Discussion on Exam 1

417T Part 2
Machine Learning Techniques

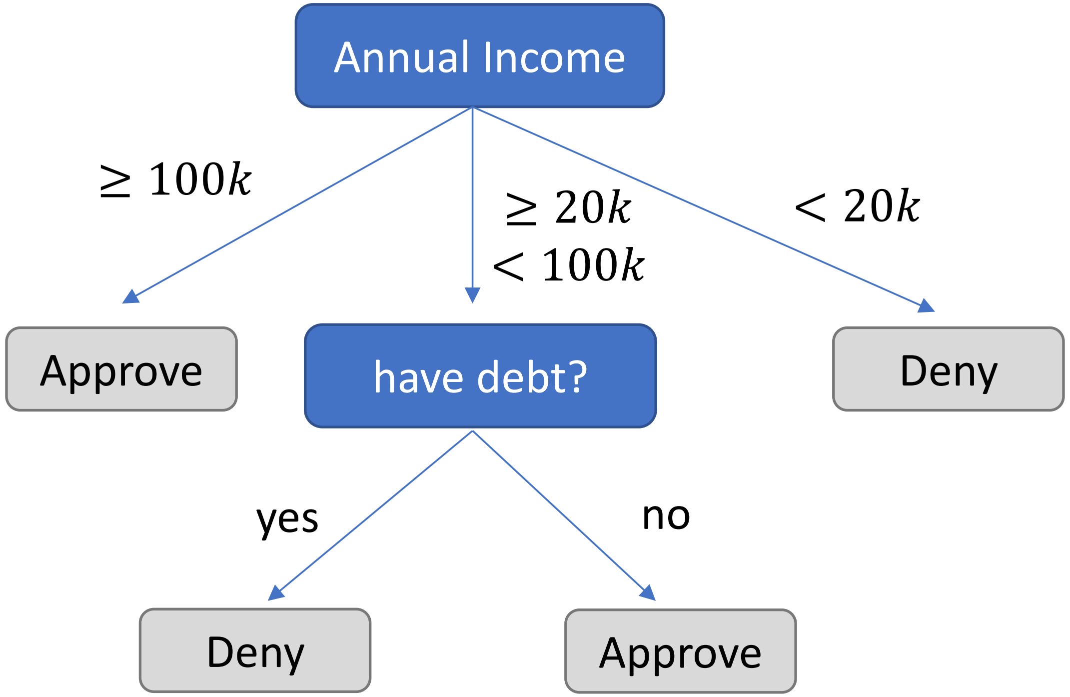

UNKNOWN TARGET DISTRIBUTION
(target function f plus noise)
$P(y \mid \mathbf{x})$
$y_n \sim P(y \mid \mathbf{x}_n)$
UNKNOWN
INPUT DISTRIBUTION
$P(\mathbf{x})$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$
$\mathbf{x}$
ERROR
MEASURE
$g(\mathbf{x}) \approx f(\mathbf{x})$
LEARNING
ALGORITHM
$\mathcal{A}$
FINAL
HYPOTHESIS
g
HYPOTHESIS SET
$\mathcal{H}$



Focus of the rest
of the semester

Decision Tree

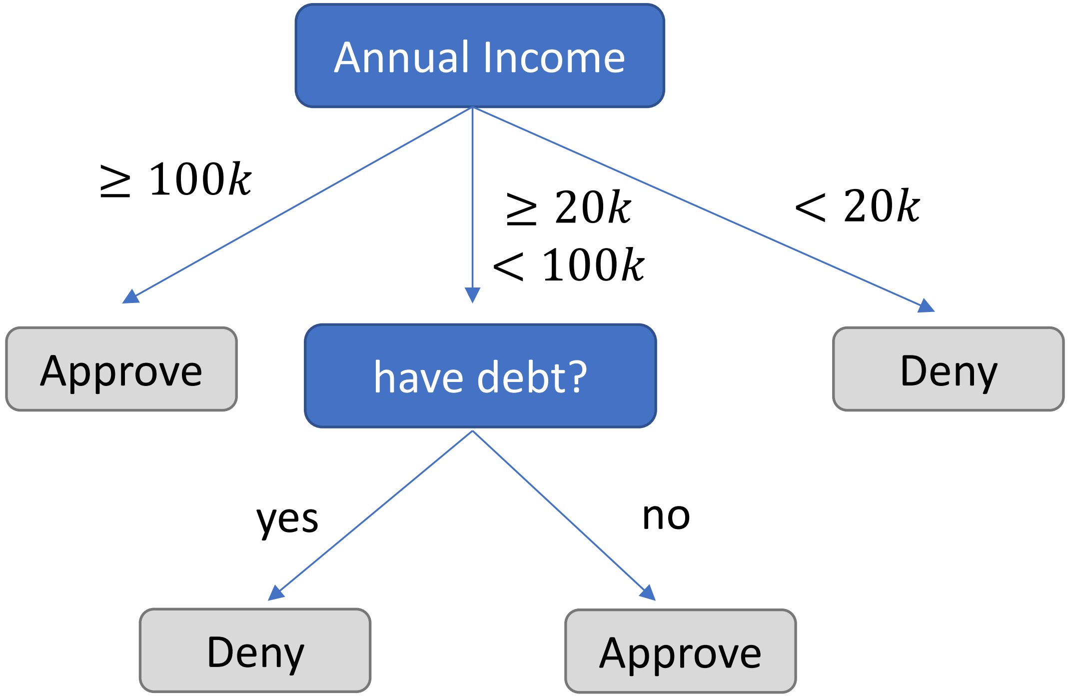
Decision Tree Hypothesis



Credit Card Approval Example

- $\vec{x} = (\text{annual income, have debt})$
- $y \in \{\text{approve, deny}\}$

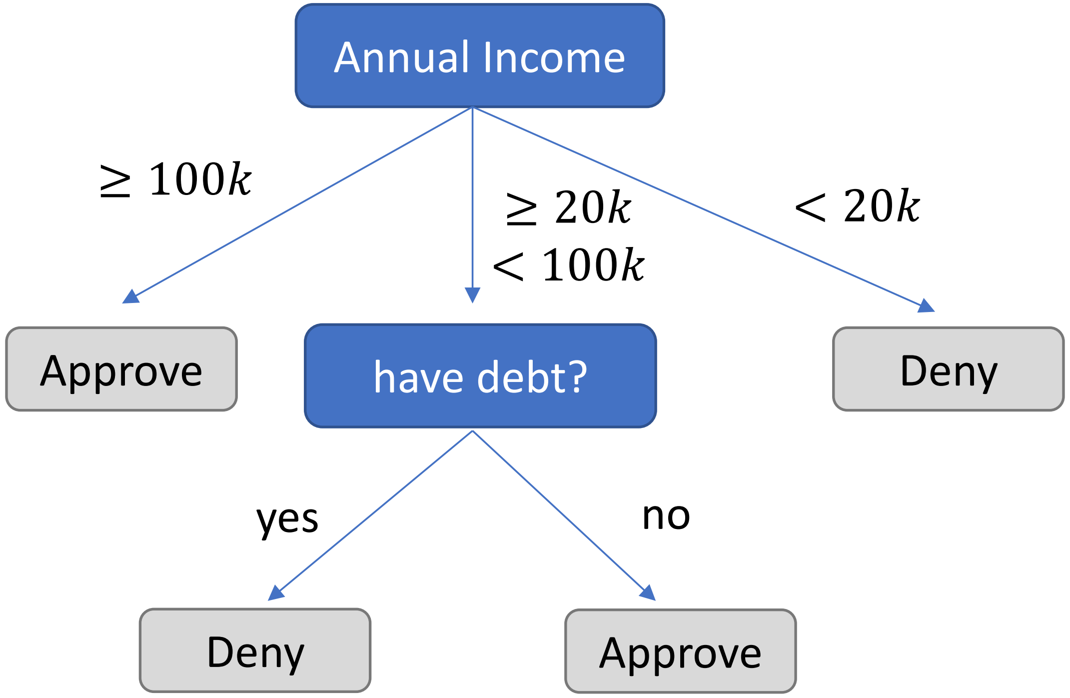
Decision Tree <u>Hypothesis</u>



Credit Card Approval Example

- Pros
  - Easy to interpret (interpretability is getting attention and is important in some domains)
  - Can handle multi-type data (Numerical, categorical. …)
  - Easy to implement (Bunch of if-else rules)
- Cons

Decision Tree Hypothesis



Credit Card Approval Example

- Pros
  - Easy to interpret (interpretability is getting attention and is important in some domains)
  - Can handle multi-type data (Numerical, categorical. …)
  - Easy to implement (Bunch of if-else rules)
- Cons
  - Generally speaking, bad generalization
  - VC dimension is infinity
  - High variance (small change of data leads to very different hypothesis)
  - Easily overfit
- Why we care?
  - One of the classical model
  - Building block for other models (e.g., random forest)

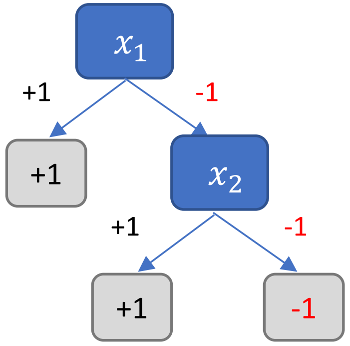
Learning Decision Tree from Data

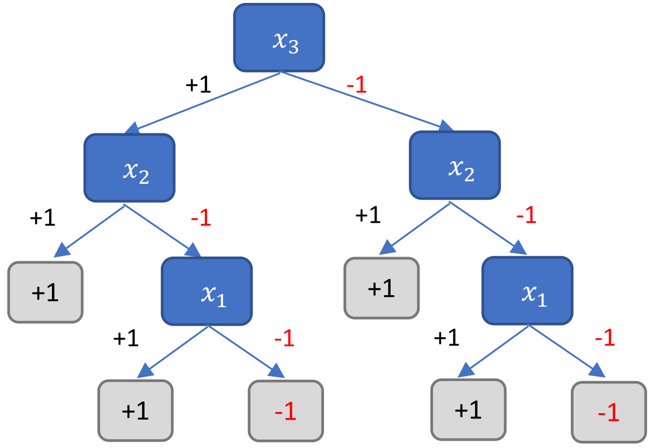
- Given dataset D, how to learn a decision tree hypothesis?

x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

- Potential approach
  - Find $g = argmin_{h \in H} E_{in}(h)$

- Multiple decision trees with zero E_{in}





Which one do you think might generalize better?

Learning Decision Tree from Data

- Conceptual intuition to deal with overfitting
  - Regularization: Constrain H

- Informally,

minimize $E_{in}(\vec{w})$

subject to $size(tree) \le C$

- This optimization is generally computationally intractable.
- Most decision tree learning algorithms rely on ***heuristics*** to approximate the goal.

Greedy-Based Decision Tree Algorithm

- DecisionTreeLearn(D): Input a dataset D, output a decision tree hypothesis
  - Create a root node r
  - If termination conditions are met
    - return a single node tree with leaf prediction based on D
  - Else: Greedily find a feature A to split according to split criteria
    - For each possible value v_i of A
      - Let D_i be the dataset containing data with value v_i for feature A
      - Create a subtree DecisionTreeLearn(D_i) that being the child of root r

- Most decision tree learning algorithms follow this template, but with different choices of heuristics

Example

DecisionTreeLearn(D)
Create a root node r
If termination conditions are met
return a single node tree with leaf prediction based on
Else: Greedily find a feature A to split according to split criteria
For each possible value v_i of A
Let D_i be the dataset containing data with value v_i for feature A
Create a subtree DecisionTreeLearn(D_i) that being the child of root r

x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

DecisionTreeLearn

Termination conditions not net
Find a feature to split

x_1

+1 -1

x_1	x_2	x_3	y
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1

DecisionTreeLearn

terminate

Leaf prediction +1

x_1	x_2	x_3	y
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

DecisionTreeLearn

Don't terminate

Find next feature to split

Example Heuristics

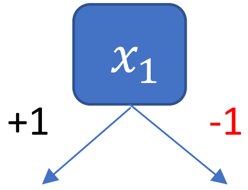
DecisionTreeLearn(D)
Create a root node r
If termination conditions are met
return a single node tree with leaf prediction based on
Else: Greedily find a feature A to split according to split criteria
For each possible value v_i of A
Let D_i be the dataset containing data with value v_i for feature A
Create a subtree DecisionTreeLearn(D_i) that being the child of root r

- Termination conditions
  - When the dataset is empty
  - When all labels are the same
  - when all features are the same
  - When the depth of the tree is too deep
  - …
- Leaf predictions
  - Majority voting
  - Average (for regression)
  - …
- Split criteria?

Split Criteria

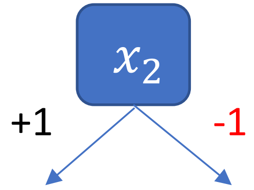
- Which feature would you choose to split?

x_1	x_2	y
+1	+1	+1
+1	-1	+1
-1	+1	-1
-1	-1	-1



x_1	x_2	y
+1	+1	+1
+1	-1	+1

x_1	x_2	y
-1	+1	-1
-1	-1	-1



x_1	x_2	y
+1	+1	+1
-1	+1	-1

x_1	x_2	y
+1	-1	+1
-1	-1	-1

- Want the tree to be "smaller"
  - Intuition: choose the one that the labels are more "pure"
  - Example: choose the one maximizing information gain => ID3 Algorithm

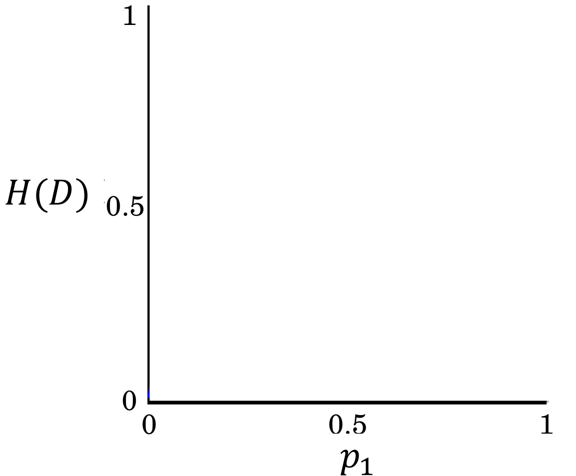
Brief Intro to Information Entropy

- Assume there are K possible labels
- Entropy:
  - $H(D) = \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i}$
  - p_i: ratio of points with label i in the data

By definition

$$0 \log_2 \frac{1}{0} = 0\,;\ 1 \log_2 \frac{1}{1} = 0$$

- Binary case with $K = 2$



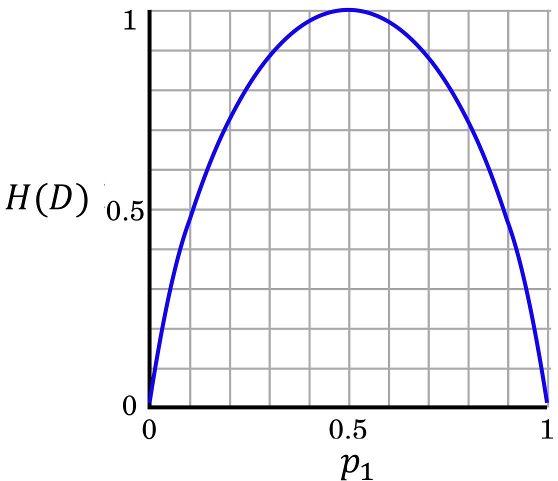
Brief Intro to Information Entropy

- Assume there are K possible labels
- Entropy:
  - $H(D) = \sum_{i=1}^{K} p_i \log_2 \frac{1}{p_i}$
  - p_i: ratio of points with label i in the data

By definition

$0 \log_2 \frac{1}{0} = 0 \,;\; 1 \log_2 \frac{1}{1} = 0$

- Binary case with $K = 2$



- Interpretations of entropy
  - Expected # bit to encode a distribution
- Higher entropy
  - data is less "pure"
- "pure" data => all labels are +1 or -1 => entropy = 0
- Want to choose splits that lead to pure data, i.e., lower entropy

ID3: Using Information Gain as Selection Criteria

- Information gain of choosing feature A to split
  - $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i)$ [The amount of decrease in entropy]
- ID3: Choose the split that maximize $Gain(D, A)$

Notation:
$|D|$ is the number of points in D

DecisionTreeLearn(D)
    Create a root node r
    If termination conditions are met
        return a single node tree with leaf prediction based on
    Else: Greedily find a feature A to split according to split criteria
    For each possible value v_i of A
        Let D_i be the dataset containing data with value v_i for feature A
        Create a subtree DecisionTreeLearn(D_i) that being the child of root r

- ID3 termination conditions
  - If all labels are the same
  - If all features are the same
  - If dataset is empty
- ID3 leaf predictions
  - Most common labels (majority voting)
- ID3 split criteria
  - Information gain

ID3: Using Information Gain as Selection Criteria

- Information gain of choosing feature A to split
  - $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i)$
- ID3: Choose the split that maximize $Gain(D, A)$

x_1	x_2	y
+1	+1	+1
+1	-1	+1
-1	+1	-1
-1	-1	-1

$H(D) = 0.5 \log_2 2 + 0.5 \log_2 2 = 1$

x_1

+1 / -1

x_1	x_2	y
+1	+1	+1
+1	-1	+1

$H(D_{x_1=1}) = 0$

x_1	x_2	y
-1	+1	-1
-1	-1	-1

$H(D_{x_1=-1}) = 0$

$Gain(D, x_1) = 1$

x_2

+1 / -1

x_1	x_2	y
+1	+1	+1
-1	+1	-1

$H(D_{x_2=1}) = 1$

x_1	x_2	y
+1	-1	+1
-1	-1	-1

$H(D_{x_2=-1}) = 1$

$Gain(D, x_2) = 0$

ID3 will choose x_1 as the next split attribute

Further Addressing Overfitting

- More Regularization (Constrain H)
  - Do not split leaves past a fixed depth
  - Do not split leaves with fewer than c labels
  - Do not split leaves where the maximal information gain is less than τ
- Pruning (removing leaves)
  - Evaluate each split using a validation set and compare the validation error with and without that split (replacing it with the most common label at that point)
  - Use statistical test to examine whether the split is “informative” (leads to different enough subtrees)

More Discussions

- Real-valued features (continuous x)
  - Need to select threshold for branching

- Regression (continuous y)
  - Change leaf prediction: e.g., average instead of majority vote
  - Change measure for "purity" of data: e.g., squared error of data

Ensemble Learning

The focus of the next two lectures

Ensemble Learning

- Assume we are given a set of learned hypothesis
  - $g_1, g_2, \ldots, g_M$
- What can we do?
  - Use validation to pick the best one
  - What if all of them are not good enough
- Can we aggregate them?

Is Aggregation a Good Idea?

- At a 1906 country fair, ~800 people participate in a contest to guess the weight of an ox.
- Reward is given to the person with the closest guess.
- The average guess is 1,197lbs.
  The true answer is 1,198lbs.

Is Aggregation a Good Idea?

- Maybe
  - If the hypothesis is "diverse", and "in average" they seem good
- Question:
  - How do we find a set of hypothesis that are diverse and "in average" good
  - How do we aggregate the set of hypothesis
- Ensemble learning
  - Bagging – Random Forest (March 17)
  - Boosting – AdaBoost (March 19)