CS 360: Machine Learning

Sara Mathieson, Sorelle Friedler

Spring 2024

HAVERFORD
COLLEGE

Admin

- **Lab 2** due TODAY
- **Sorelle office hours** TODAY, 4-5pm in H110
- **Lab 3** released tonight (Decision Trees)


hav.to/i8j
GAME
DEVELOPMENT
MINI HACKATHON
BEGINNERS
ECOURAGED
FEB
9+10

Outline for Feb 8

- Finish Cross Validation
- Decision Tree introduction
- ID3 algorithm
- Handout 6
- Implementation suggestions

Outline for Feb 8

- Finish Cross Validation
- Decision Tree introduction
- ID3 algorithm
- Handout 6
- Implementation suggestions

Cross Validation: other considerations

- Can use cross-validation to choose hyperparameters
- Leave-one-out cross validation (LOOCV)
  - Special case of $k=n$
  - Train using $n-1$ examples, evaluate on remaining
  - Repeat n times
- Can do multiple trials of CV

Examples of parameters vs. hyperparameters

- Polynomial regression
  - Hyperparameter: degree of the polynomial
  - Parameters: weights on each feature (or power of a feature)
- Logistic regression
  - Hyperparameter: learning rate, max iterations
  - Parameters: weights on each feature
- K-nearest neighbors
  - Hyperparameters: K (number of neighbors), distance metric

Hyperparameters

- Difficult to define precisely, but typically a parameter that controls other parameters

- We can't choose hyperparameters via test data (breaks cardinal rule of not looking at our test data!)

- But we can use *validation data*

Finding hyper-parameters

- Grid search
- Random search

```python
from sklearn.model_selection import GridSearchCV

full_pipeline = Pipeline([
    ("preprocessing", preprocessing),
    ("random_forest", RandomForestRegressor(random_state=42)),
])
param_grid = [
    {'preprocessing__geo__n_clusters': [5, 8, 10],
     'random_forest__max_features': [4, 6, 8]},
    {'preprocessing__geo__n_clusters': [10, 15],
     'random_forest__max_features': [6, 8, 10]},
]
grid_search = GridSearchCV(full_pipeline, param_grid, cv=3,
                           scoring='neg_root_mean_squared_error')
grid_search.fit(housing, housing_labels)
```

n_clusters	max_features	split0	split1	split2	mean_test_rmse
15	6	43460	43919	44748	44042
15	8	44132	44075	45010	44406
15	10	44374	44286	45316	44659
10	6	44683	44655	45657	44999
10	6	44683	44655	45657	44999

The Short Way
(that Many People Actually Use)

- Split into only training data + validation data
- Train on training data, evaluate on validation data
- Report cross-validation performance
  - possibly also training performance

- Why is this used?
  - might not be enough data to create held-out test set
  - you cannot trust that authors did not peek at test data anyway =P

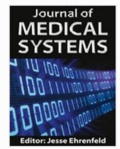
Slide: Jessica Wu

Outline for Feb 8

- Finish Cross Validation
- Decision Tree introduction
- ID3 algorithm
- Handout 6
- Implementation suggestions

Real-World Examples

- Medical diagnostics



Journal of Medical Systems
October 2002, Volume 26, Issue 5, pp 445–463 | Cite as

Decision Trees: An Overview and Their Use in Medicine

Authors   Authors and affiliations

Vili Podgorelec, Peter Kokol, Bruno Stiglic, Ivan Rozman

- Use decision trees to interpret another ML algorithm (SVMs)

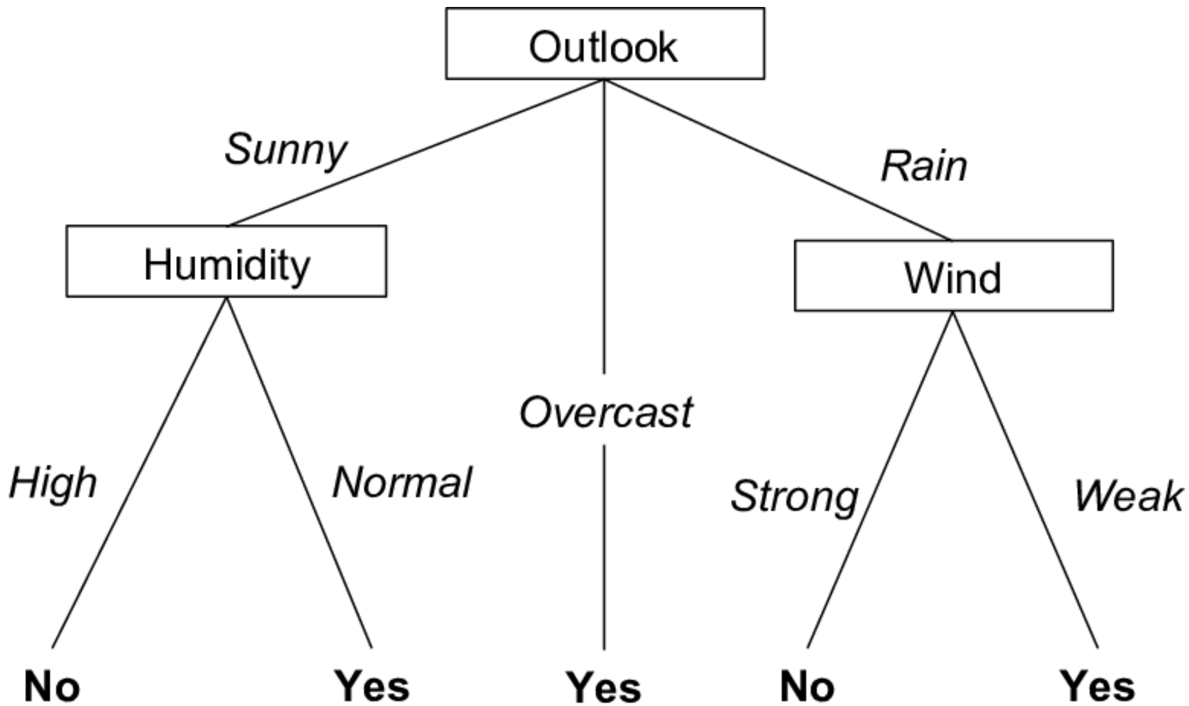
Machine-learning-assisted materials discovery using failed experiments

Paul Raccuglia, Katherine C. Elbert, Philip D. F. Adler, Casey Falk, Malia B. Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A. Friedler, Joshua Schrier & Alexander J. Norquist

Nature **533**, 73–76 (05 May 2016) | Download Citation

Optional Reading!

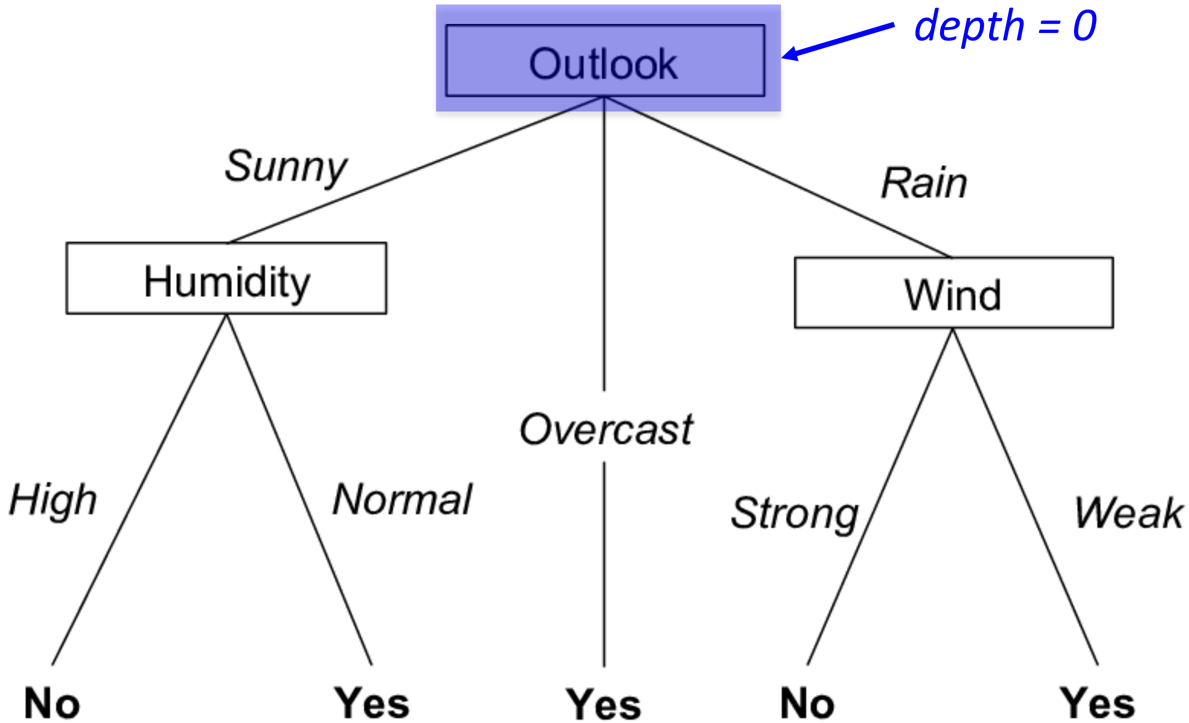
Decision Tree example (tennis data)



- Each internal node: test one feature
- Each branch from node: selects one value of the feature
- Each leaf node: predict *y*

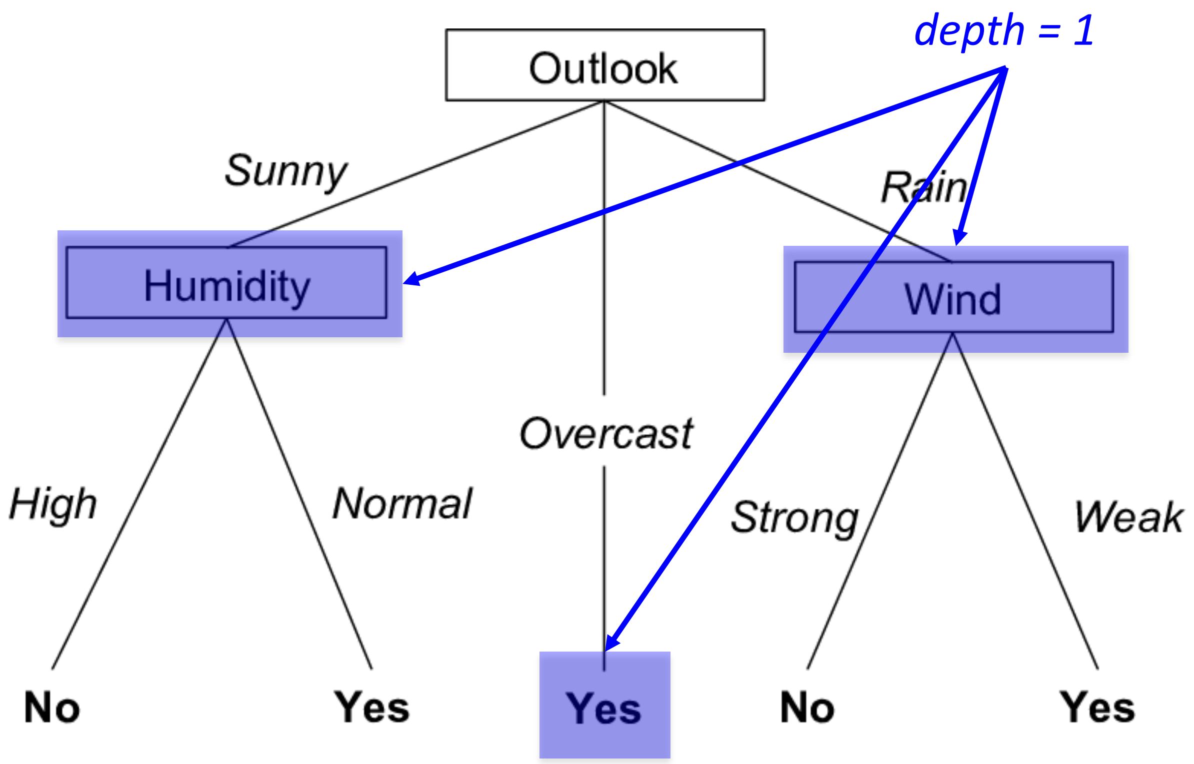
Based on slides by Jessica Wu and Eric Eaton [originally by Tom Mitchell]

Decision Tree example (tennis data)



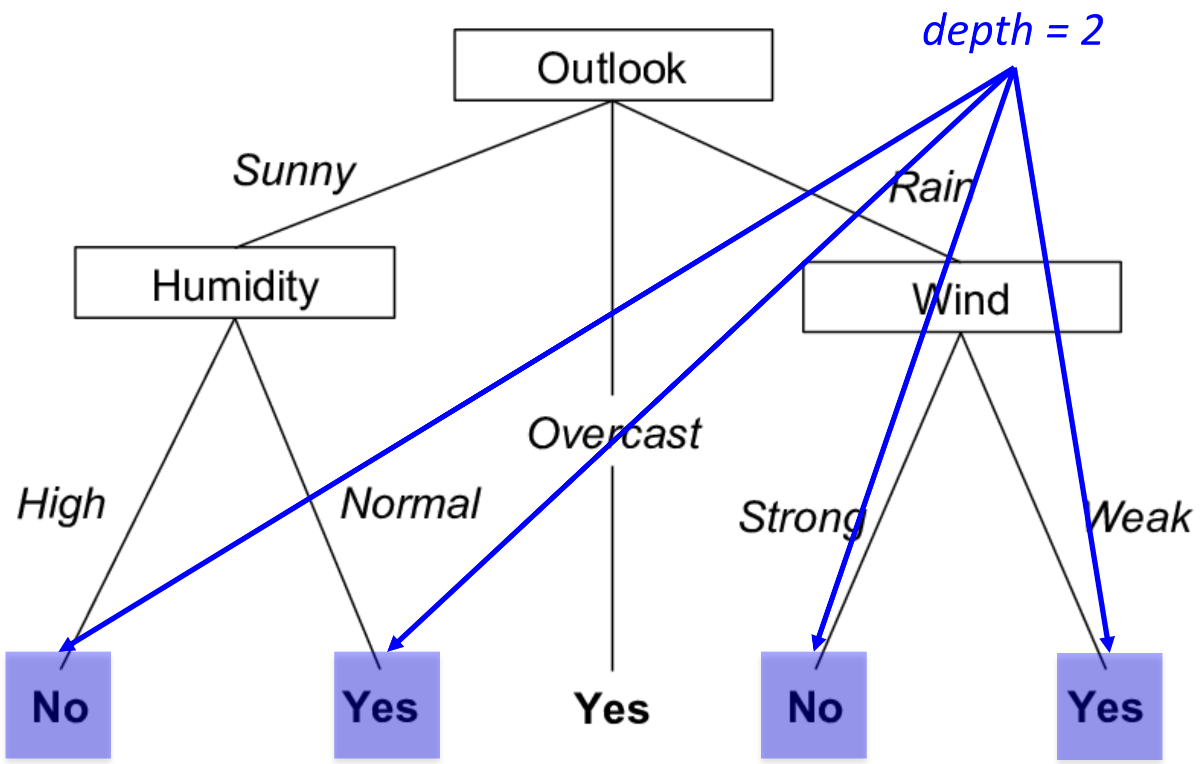
Key term: *depth*

Decision Tree example (tennis data)



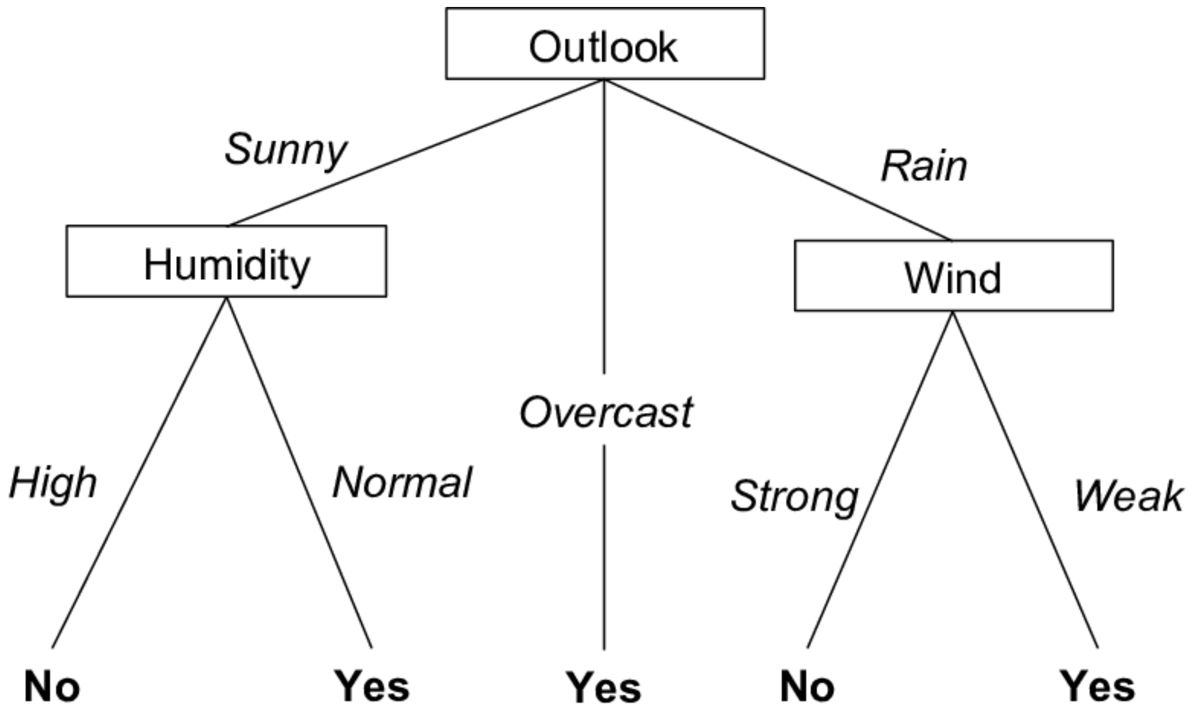
Key term: *depth*

Decision Tree example (tennis data)



Key term: *depth*

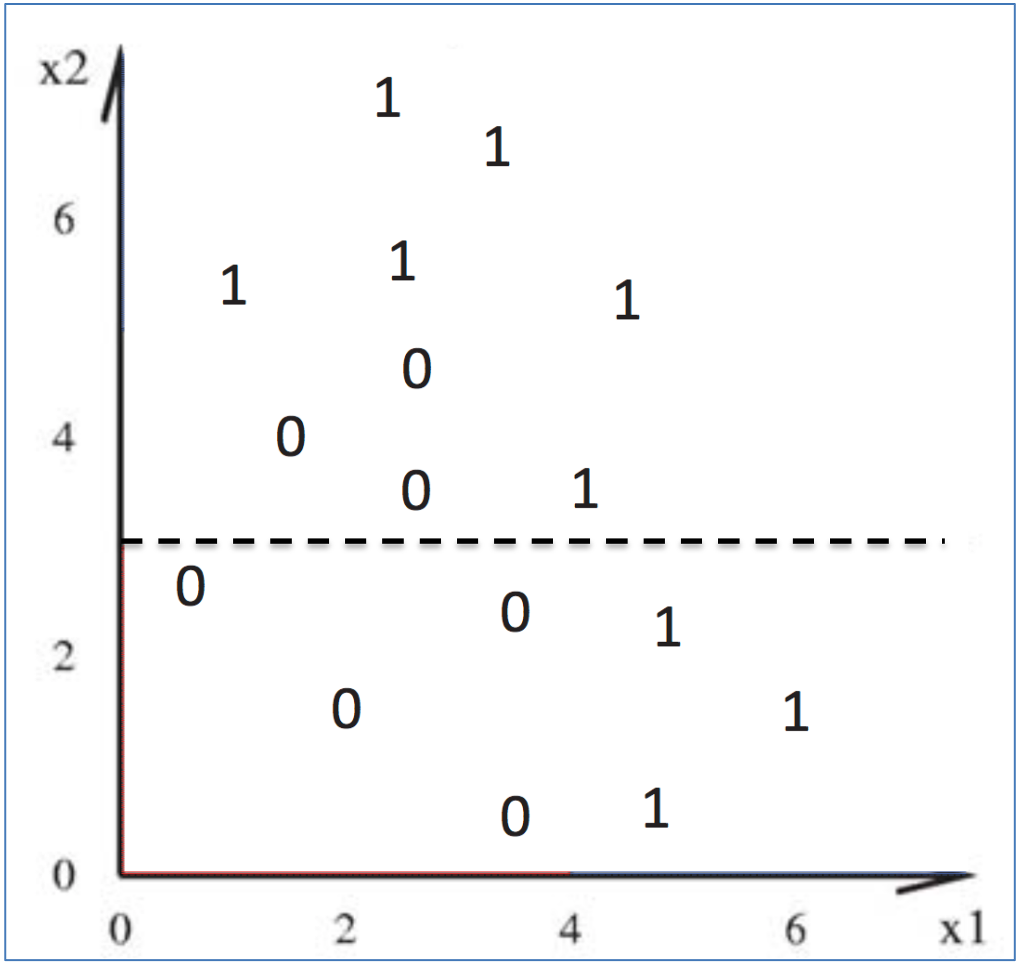
Decision Tree example (tennis data)



	Outlook	Temp	Humidity	Wind
(test example) ***x*** =	Rain	Hot	High	Strong

y_{pred} = No

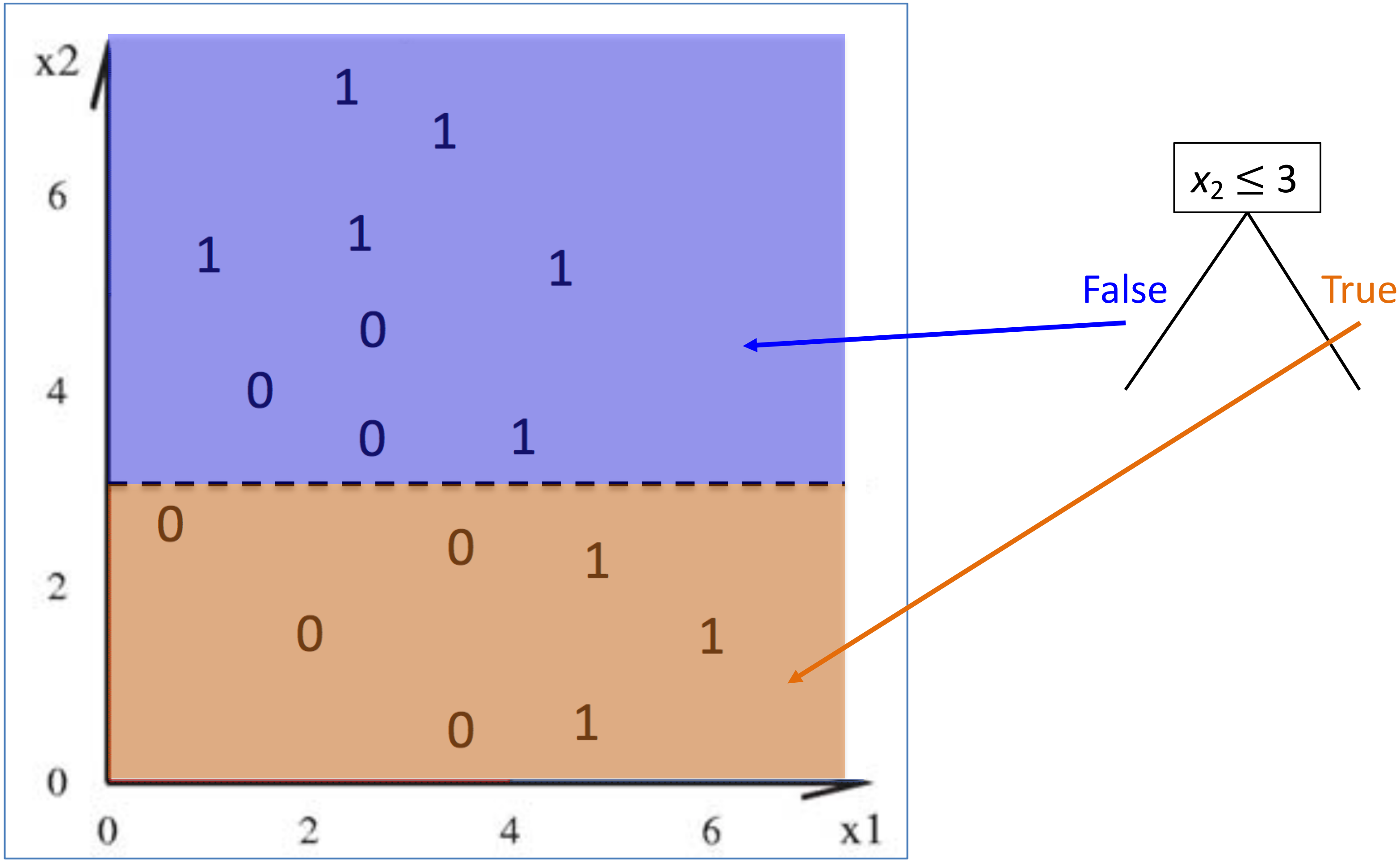
Can also consider continuous features



$x_2 \leq 3$

Example by: Eric Eaton

Can also consider continuous features



Example by: Eric Eaton

Decision Tree pros/cons

- Very interpretable! Easy to say *why* we made a classification (can point to which features)
- Compact representation and fast predictions

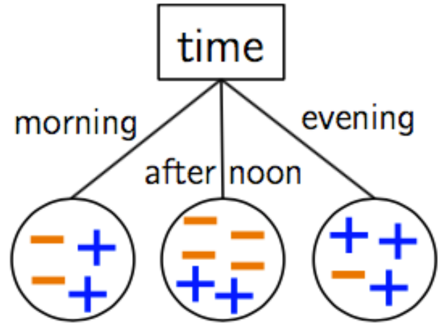
- Can be brittle (not looking at each example holistically)
- Featurization and implementation difficulties

Check-in: work individually for a few minutes

1. Match the decision tree component on the left with its corresponding data component on the right.

• internal nodes	class labels
• branches	feature names
• leaves	feature values

2. Say I am trying to predict if a student will like a course (+) or dislike it (−). One of the features is the time of day the course is offered. If I just choose this one feature and build a decision tree, here is how the training examples cluster at the leaves:



   (a) How would you classify a new example with value **evening** for the feature **time**?

   (b) What is the overall *training error* if I use the majority class label at each leaf?

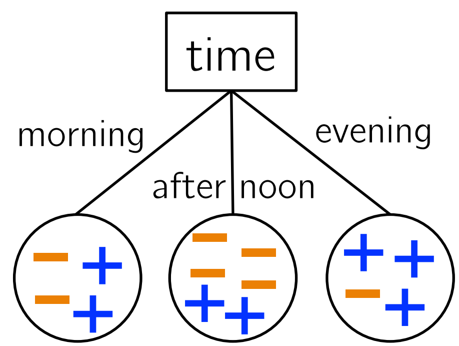
3. If a decision tree is overfitting, is the *depth* more likely to be low or high?

Check-in

1)
- internal nodes — class labels
- branches — feature names
- leaves — feature values

2) (a) +

(b) 5/14



3) high

Outline for Feb 8

- Finish Cross Validation
- Decision Tree introduction
- ID3 algorithm
- Handout 6
- Implementation suggestions

ID3 Decision Tree algorithm (1986)

- Select feature that "best" informs label prediction (i.e. *y*)
- **Divide**: partition data into branches based on their value at this feature
- **Conquer**: recurse on each partition

Optional reading

Machine Learning 1: 81–106, 1986

Induction of Decision Trees

J.R. QUINLAN (munnari!nswitgould.oz!quinlan@seismo.css.gov)
Centre for Advanced Computing Sciences, New South Wales Institute of Technology, Sydney 2007, Australia

(Received August 1, 1985)

Key words: classification, induction, decision trees, information theory, knowledge acquisition, expert systems

Top-Down decision tree algorithm

Dataset (X,y)

Features

MakeSubtree(*D*, *F*)
  if stopping criteria met
    make a leaf node *N*
    determine class label/probabilities for *N*

For us: use majority label
(break ties arbitrarily)

Top-Down decision tree algorithm

Dataset (X,y)

Features

MakeSubtree(*D*, *F*)
  if stopping criteria met
    make a leaf node *N*
    determine class label/probabilities for *N*
  else
    make an internal node *N*
    S = FindBestFeature(*D*, *F*)
    for each outcome *k* of *S*
      D_k = subset of instances that have outcome *k*
      N.child[k] = MakeSubtree(D_k,*F*-*S*)
  return subtree rooted at *N*

For us: use majority label
(break ties arbitrarily)

Why don't we want to use this feature again?

Design choice: stopping criteria

1. All the data points in our partition have the same label

2. No more features remain to split on

3. No features are informative about the label

   i.e. all have same remaining features but there is still label heterogeneity

4. Reached (user specified) max depth in the tree

   For our Lab 3 implementation

Additional base case options

- Stop when leaf label reaches a certain fraction (i.e. 95% “yes”, 5% “no”)

- Set a minimum number of examples in leaf (i.e. if we have a 2-1 split, stop)

Outline for Feb 8

- Finish Cross Validation
- Decision Tree introduction
- ID3 algorithm
- Handout 6
- Implementation suggestions

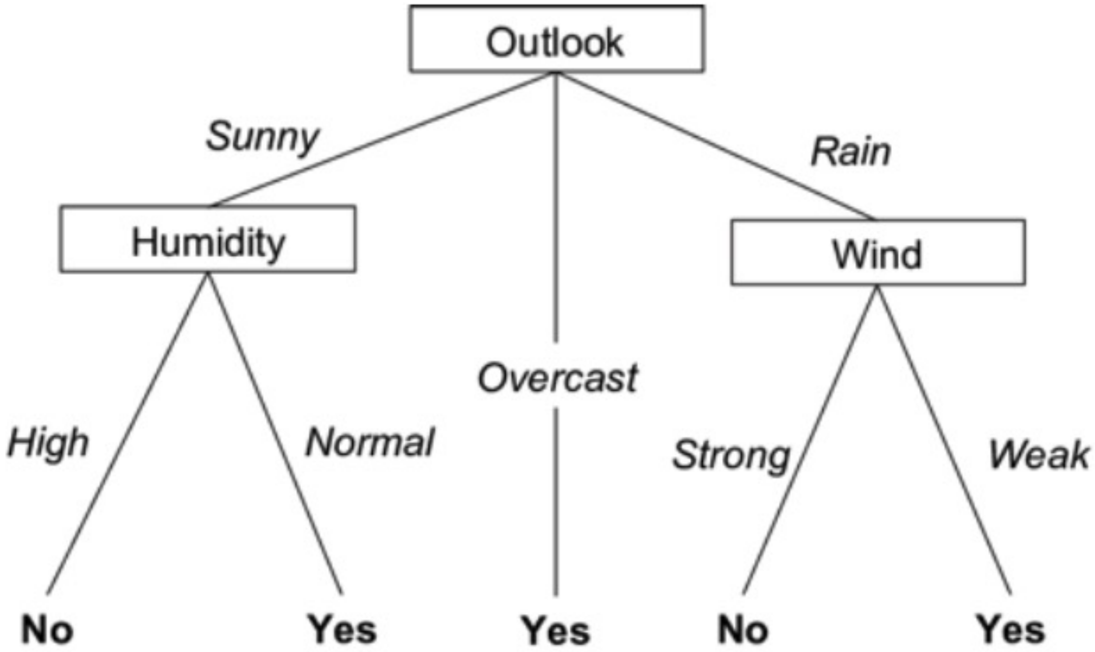
Handout 6

1. First, what is n (number of data points)? What is p (number of features)? Given the training data and decision tree shown below, what is the classification error on this data?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (y)
$\boldsymbol{x}_1$	Sunny	Hot	High	Weak	No
$\boldsymbol{x}_2$	Sunny	Hot	High	Strong	No
$\boldsymbol{x}_3$	Overcast	Hot	High	Weak	Yes
$\boldsymbol{x}_4$	Rain	Mild	High	Weak	Yes
$\boldsymbol{x}_5$	Rain	Cool	Normal	Weak	Yes
$\boldsymbol{x}_6$	Rain	Cool	Normal	Strong	No
$\boldsymbol{x}_7$	Overcast	Cool	Normal	Strong	Yes
$\boldsymbol{x}_8$	Sunny	Mild	High	Weak	No
$\boldsymbol{x}_9$	Sunny	Cool	Normal	Weak	Yes
$\boldsymbol{x}_{10}$	Rain	Mild	Normal	Weak	Yes
$\boldsymbol{x}_{11}$	Sunny	Mild	Normal	Strong	Yes
$\boldsymbol{x}_{12}$	Overcast	Mild	High	Strong	Yes
$\boldsymbol{x}_{13}$	Overcast	Hot	Normal	Weak	Yes
$\boldsymbol{x}_{14}$	Rain	Mild	High	Strong	No

Handout 6

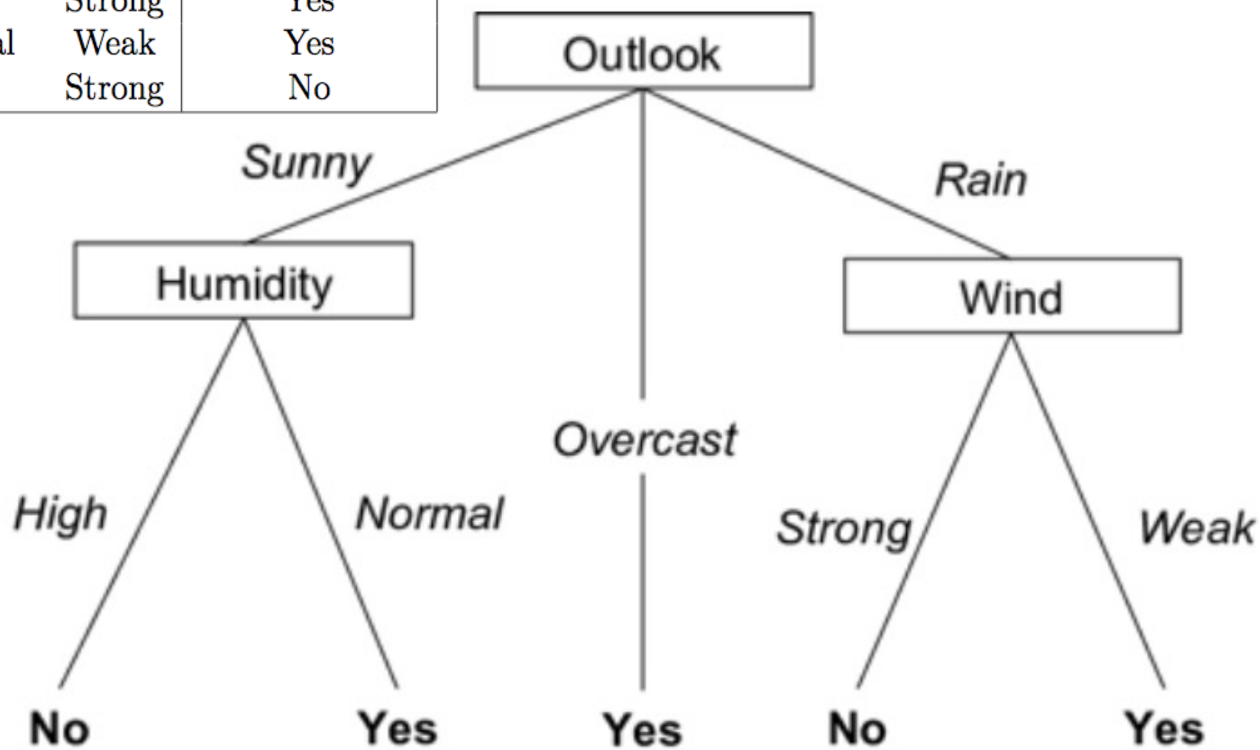
2. On the tree below, the children of each node divide the training data into partitions. Label each node (both internal nodes and leaves) with the counts of "No" and "Yes" labels based on the partition. For example, the counts for the node labeled *Outlook* would be [5, 9].



3. What if we had restricted the tree's *depth* to be 1? What would the tree look like and what would be the classification error?

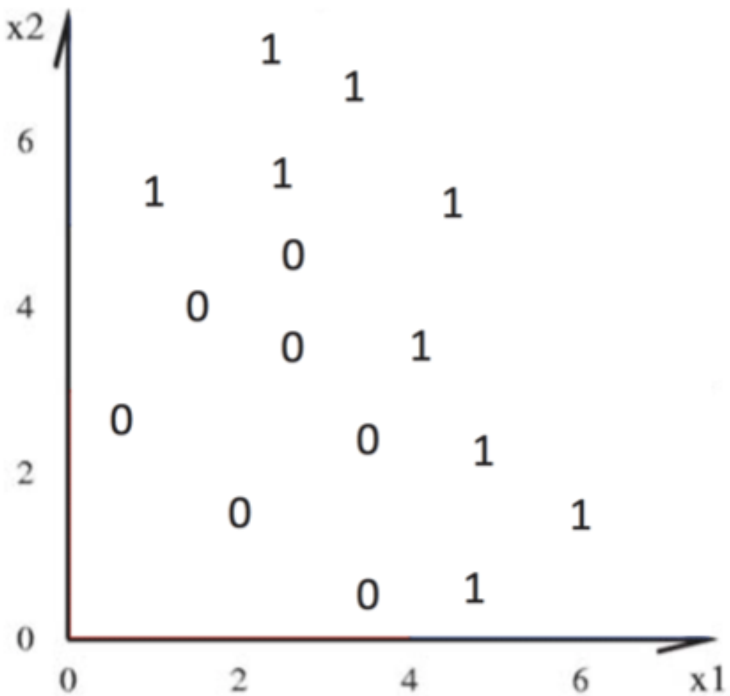
Handout 6

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (y)
$\boldsymbol{x}_1$	Sunny	Hot	High	Weak	No
$\boldsymbol{x}_2$	Sunny	Hot	High	Strong	No
$\boldsymbol{x}_3$	Overcast	Hot	High	Weak	Yes
$\boldsymbol{x}_4$	Rain	Mild	High	Weak	Yes
$\boldsymbol{x}_5$	Rain	Cool	Normal	Weak	Yes
$\boldsymbol{x}_6$	Rain	Cool	Normal	Strong	No
$\boldsymbol{x}_7$	Overcast	Cool	Normal	Strong	Yes
$\boldsymbol{x}_8$	Sunny	Mild	High	Weak	No
$\boldsymbol{x}_9$	Sunny	Cool	Normal	Weak	Yes
$\boldsymbol{x}_{10}$	Rain	Mild	Normal	Weak	Yes
$\boldsymbol{x}_{11}$	Sunny	Mild	Normal	Strong	Yes
$\boldsymbol{x}_{12}$	Overcast	Mild	High	Strong	Yes
$\boldsymbol{x}_{13}$	Overcast	Hot	Normal	Weak	Yes
$\boldsymbol{x}_{14}$	Rain	Mild	High	Strong	No



Handout 6: continuous features

4. For the dataset below, the label $y \in \{0, 1\}$. What is n? What is p? Devise a decision tree for this data that perfectly classifies the given examples. Internal node labels should be of the form "$x_j \leq a$", where a is some constant.



5. Repeat Question (2) for this decision tree (i.e. label each node with the "0" and "1" counts.)

Outline for Feb 8

- Finish Cross Validation
- Decision Tree introduction
- ID3 algorithm
- Handout 6
- Implementation suggestions

Recursive algorithm: Partition data structure

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (y)
$\boldsymbol{x}_1$	Sunny	Hot	High	Weak	No
$\boldsymbol{x}_2$	Sunny	Hot	High	Strong	No
$\boldsymbol{x}_3$	Overcast	Hot	High	Weak	Yes
$\boldsymbol{x}_4$	Rain	Mild	High	Weak	Yes
$\boldsymbol{x}_5$	Rain	Cool	Normal	Weak	Yes
$\boldsymbol{x}_6$	Rain	Cool	Normal	Strong	No
$\boldsymbol{x}_7$	Overcast	Cool	Normal	Strong	Yes
$\boldsymbol{x}_8$	Sunny	Mild	High	Weak	No
$\boldsymbol{x}_9$	Sunny	Cool	Normal	Weak	Yes
$\boldsymbol{x}_{10}$	Rain	Mild	Normal	Weak	Yes
$\boldsymbol{x}_{11}$	Sunny	Mild	Normal	Strong	Yes
$\boldsymbol{x}_{12}$	Overcast	Mild	High	Strong	Yes
$\boldsymbol{x}_{13}$	Overcast	Hot	Normal	Weak	Yes
$\boldsymbol{x}_{14}$	Rain	Mild	High	Strong	No

Recursive algorithm: Partition data structure

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (y)
$\boldsymbol{x}_1$	Sunny	Hot	High	Weak	No
$\boldsymbol{x}_2$	Sunny	Hot	High	Strong	No
$\boldsymbol{x}_3$	Overcast	Hot	High	Weak	Yes
$\boldsymbol{x}_4$	Rain	Mild	High	Weak	Yes
$\boldsymbol{x}_5$	Rain	Cool	Normal	Weak	Yes
$\boldsymbol{x}_6$	Rain	Cool	Normal	Strong	No
$\boldsymbol{x}_7$	Overcast	Cool	Normal	Strong	Yes
$\boldsymbol{x}_8$	Sunny	Mild	High	Weak	No
$\boldsymbol{x}_9$	Sunny	Cool	Normal	Weak	Yes
$\boldsymbol{x}_{10}$	Rain	Mild	Normal	Weak	Yes
$\boldsymbol{x}_{11}$	Sunny	Mild	Normal	Strong	Yes
$\boldsymbol{x}_{12}$	Overcast	Mild	High	Strong	Yes
$\boldsymbol{x}_{13}$	Overcast	Hot	Normal	Weak	Yes
$\boldsymbol{x}_{14}$	Rain	Mild	High	Strong	No

$\boldsymbol{x}_1$	Sunny	Hot	High	Weak	No
$\boldsymbol{x}_2$	Sunny	Hot	High	Strong	No
$\boldsymbol{x}_8$	Sunny	Mild	High	Weak	No
$\boldsymbol{x}_9$	Sunny	Cool	Normal	Weak	Yes
$\boldsymbol{x}_{11}$	Sunny	Mild	Normal	Strong	Yes

Recursive algorithm: Partition data structure

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (y)
x_1	Sunny	Hot	High	Weak	No
x_2	Sunny	Hot	High	Strong	No
x_3	Overcast	Hot	High	Weak	Yes
x_4	Rain	Mild	High	Weak	Yes
x_5	Rain	Cool	Normal	Weak	Yes
x_6	Rain	Cool	Normal	Strong	No
x_7	Overcast	Cool	Normal	Strong	Yes
x_8	Sunny	Mild	High	Weak	No
x_9	Sunny	Cool	Normal	Weak	Yes
x_{10}	Rain	Mild	Normal	Weak	Yes
x_{11}	Sunny	Mild	Normal	Strong	Yes
x_{12}	Overcast	Mild	High	Strong	Yes
x_{13}	Overcast	Hot	Normal	Weak	Yes
x_{14}	Rain	Mild	High	Strong	No

x_1	Sunny	Hot	High	Weak	No
x_2	Sunny	Hot	High	Strong	No
x_8	Sunny	Mild	High	Weak	No
x_9	Sunny	Cool	Normal	Weak	Yes
x_{11}	Sunny	Mild	Normal	Strong	Yes

x_3	Overcast	Hot	High	Weak	Yes
x_7	Overcast	Cool	Normal	Strong	Yes
x_{12}	Overcast	Mild	High	Strong	Yes
x_{13}	Overcast	Hot	Normal	Weak	Yes

Recursive algorithm: Partition data structure

Day	Outlook	Temperature	Humidity	Wind	PlayTennis (y)
x_1	Sunny	Hot	High	Weak	No
x_2	Sunny	Hot	High	Strong	No
x_3	Overcast	Hot	High	Weak	Yes
x_4	Rain	Mild	High	Weak	Yes
x_5	Rain	Cool	Normal	Weak	Yes
x_6	Rain	Cool	Normal	Strong	No
x_7	Overcast	Cool	Normal	Strong	Yes
x_8	Sunny	Mild	High	Weak	No
x_9	Sunny	Cool	Normal	Weak	Yes
x_{10}	Rain	Mild	Normal	Weak	Yes
x_{11}	Sunny	Mild	Normal	Strong	Yes
x_{12}	Overcast	Mild	High	Strong	Yes
x_{13}	Overcast	Hot	Normal	Weak	Yes
x_{14}	Rain	Mild	High	Strong	No

x_1	Sunny	Hot	High	Weak	No
x_2	Sunny	Hot	High	Strong	No
x_8	Sunny	Mild	High	Weak	No
x_9	Sunny	Cool	Normal	Weak	Yes
x_{11}	Sunny	Mild	Normal	Strong	Yes

x_3	Overcast	Hot	High	Weak	Yes
x_7	Overcast	Cool	Normal	Strong	Yes
x_{12}	Overcast	Mild	High	Strong	Yes
x_{13}	Overcast	Hot	Normal	Weak	Yes

x_4	Rain	Mild	High	Weak	Yes
x_5	Rain	Cool	Normal	Weak	Yes
x_6	Rain	Cool	Normal	Strong	No
x_{10}	Rain	Mild	Normal	Weak	Yes
x_{14}	Rain	Mild	High	Strong	No

Partition class

```python
class Example:

    def __init__(self, features, label):
        """Helper class (like a struct) that stores info about each example."""
        # dictionary. key=feature name: value=feature value for this example
        self.features = features
        self.label = label # in {-1, 1}

class Partition:

    def __init__(self, data, F):
        """Store information about a dataset"""
        self.data = data # list of examples
        # dictionary. key=feature name: value=set of possible values
        self.F = F
        self.n = len(self.data)
```

Movie	Type	Length	Director	Famous actors	Liked?
m1	Comedy	Short	Adamson	No	Yes
m2	Animated	Short	Lasseter	No	No
m3	Drama	Medium	Adamson	No	Yes
m4	Animated	Long	Lasseter	Yes	No
m5	Comedy	Long	Lasseter	Yes	No
m6	Drama	Medium	Singer	Yes	Yes
m7	Animated	Short	Singer	No	Yes
m8	Comedy	Long	Adamson	Yes	Yes
m9	Drama	Medium	Lasseter	No	Yes

How to choose the best feature? Entropy!

$\mathrm{P(Li = yes)} =$ 2/3

$\mathrm{H(Li)} =$ 0.92

$\mathrm{H(Li \mid T) = 0.61}$

$\mathrm{H(Li \mid Le) = 0.61}$

$\boxed{\mathrm{H(Li \mid D) = 0.36}}$ MIN ENTROPY

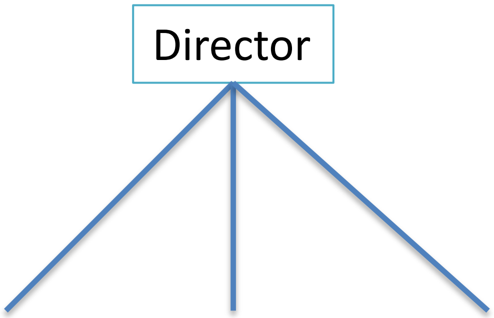
$\mathrm{H(Li \mid F) = 0.85}$

$\mathrm{Gain(Li, T)} =$ 0.92 – 0.61 = 0.31

$\mathrm{Gain(Li, Le)} =$ 0.92 – 0.61 = 0.31

$\boxed{\mathrm{Gain(Li, D)} = 0.92 - 0.36 = 0.56}$ MAX INFO GAIN

$\mathrm{Gain(Li, F)} =$ 0.92 – 0.85 = 0.07



Start of the tree

Implementation Suggestions

- Think back to trees in data structures
- Distinguish between data (X,y) and options for data (values for each feature, classes for y)

Implementation Suggestions

- Make sure you can accommodate more than two children (i.e. not a binary tree)

- Make sure your prediction/classification algorithm is recursive

- You can parse the feature name to figure out continuous/discrete and how to classify

```
age<=44.5
```

Continuous Features

(do this for the TRAIN only!)

X	Y
10	Y
7	Y
8	N
3	Y
7	N
12	Y
2	Y

1) Sort examples based on given feature

2	3	7	7	8	10	12
Y	Y	Y	N	N	Y	Y

2) Different label with same feature value, collapse to “None”

2	3	7	8	10	12
Y	Y	None	N	Y	Y

1) Whenever label changes, make a feature (use avg)

2	3	7	8	10	12
Y	Y	None	N	Y	Y

X <= 7.5

X <= 5

X <= 9