Machine Learning

Model and Learning Algorithm Evaluation

Dan Goldwasser

dgoldwas@purdue.edu

Goal for Today's class

- Decision trees wrap up and evaluation

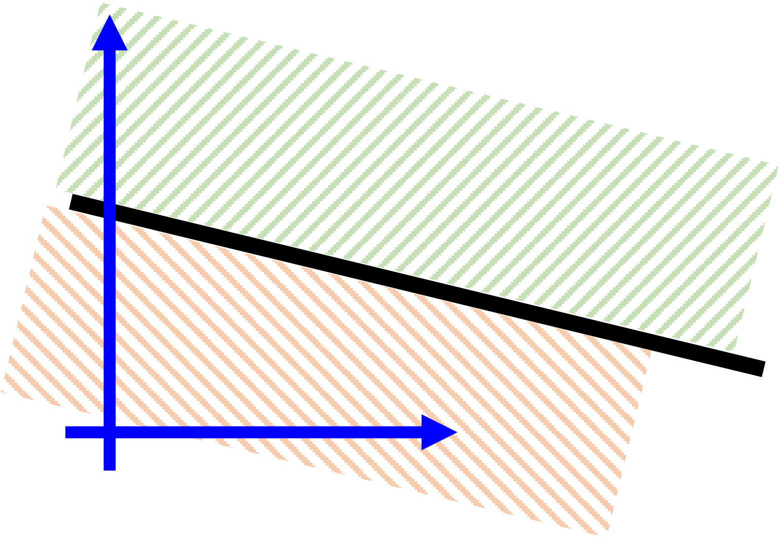
- DT learning algorithm in practice
- Overfitting vs. Underfitting in DT
- Model selection

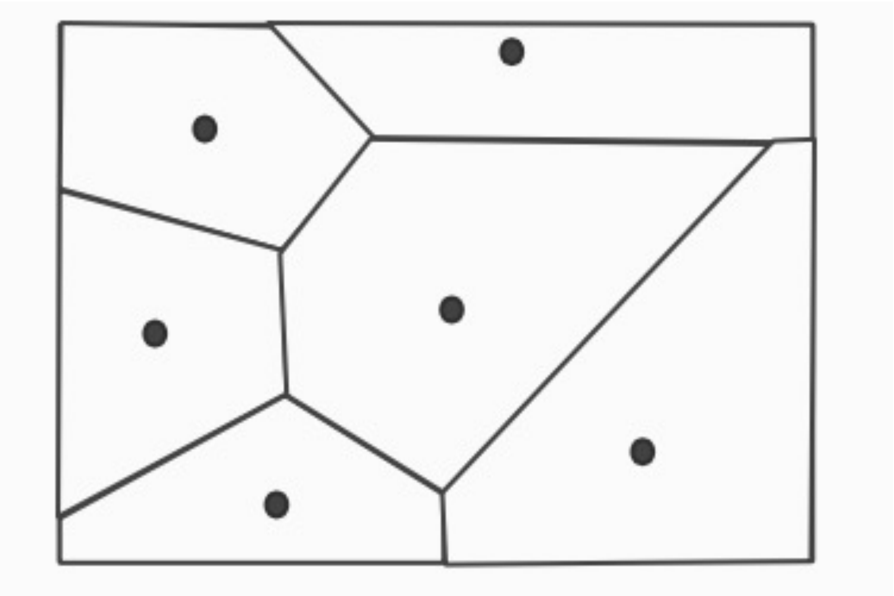
Inductive Bias

- Unbiased learning is impossible!
- Inductive bias: a set of assumptions guiding learning **beyond the data**
- ***Preference for simpler functions!***
- We have seen one type of bias: **Language bias**
  - **Pick the right hypothesis space**
- Today, as part of our discussion on Decision Trees we will introduce a second type – **search bias.**
  - **Similar objective**: Encode assumptions about learning, restrict the complexity of the resulting hypothesis

Quick Review: Expressivity

- KNN Can learn very complex decisions models
- We can try to characterize the learned function using its ***decision boundary***
  - *Visualize which elements will be classified as positive/negative*
  - *Decision Boundary is the curve separating the negative and positive regions*

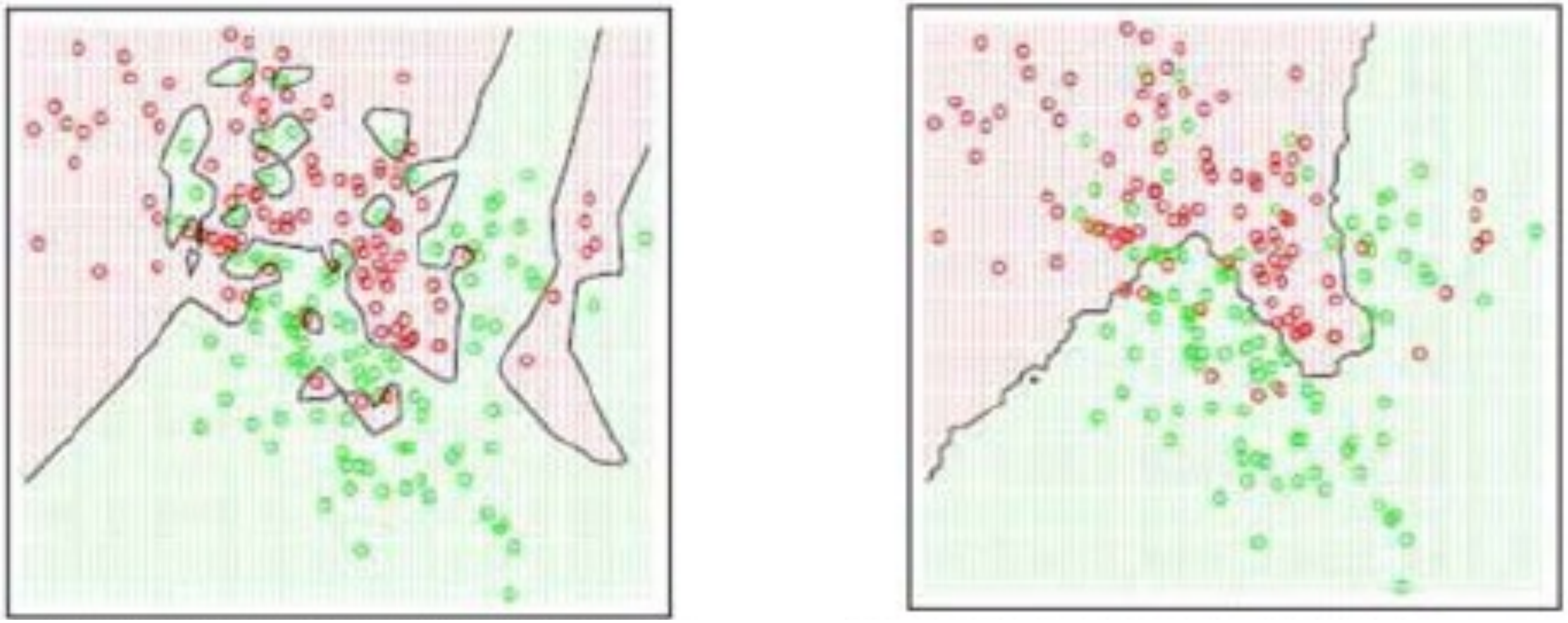




Quick Review: Expressivity

Let's take a closer look at the learned function

→ ***High sensitivity to noise!***

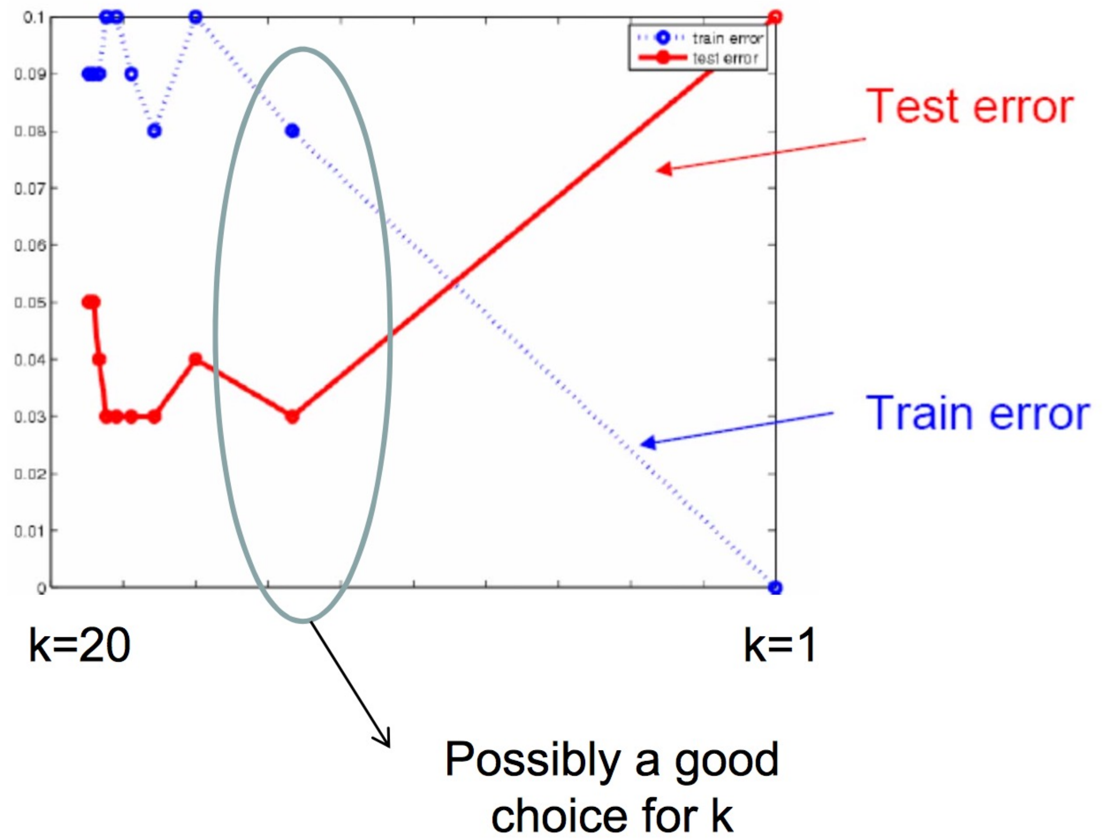


Higher k values results in smoother decision boundaries

Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

How should we set the value of K?

How would the test and train error change with K?



In general – using the training error to tune parameters will always result in a more complex hypothesis! **(why?)**

Learning Decision Trees

- KNN only stored the data, Decision trees store a "*compressed*" dataset.
  - Simplified view of DT Learning : ***better Compression, with less information loss = better generalization.***

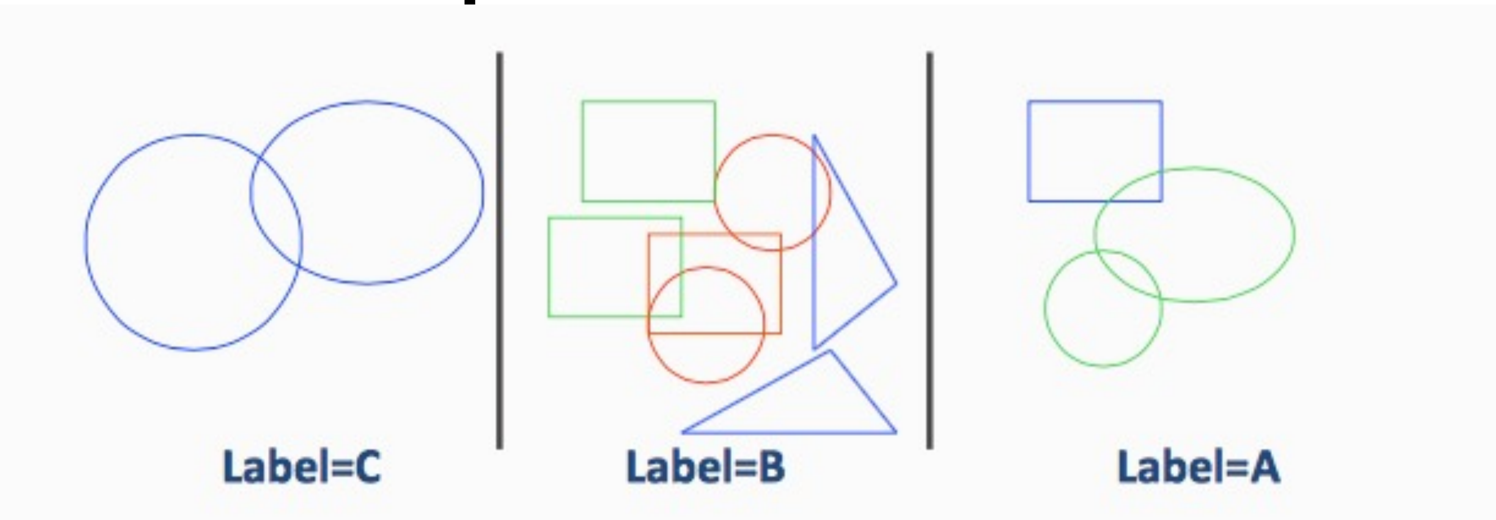
- **DT Learning overview:**
  - Decision Tree Representation
  - Algorithms for learning decision trees
  - Experimental issues
    - Controlling overfitting

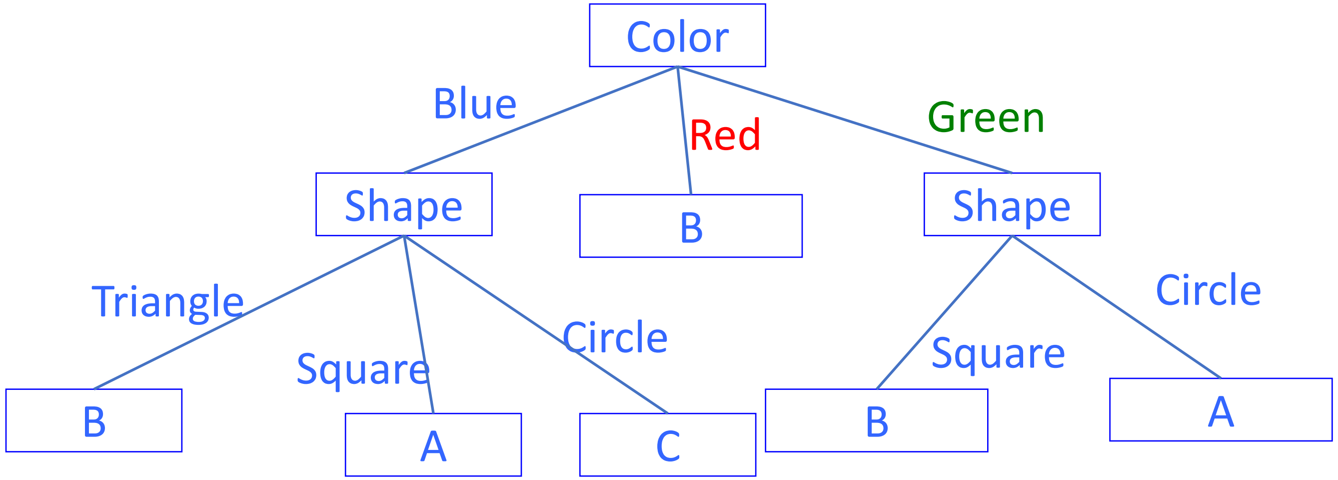
Decision Trees

- A hierarchical data structure (tree) that represents data by implementing a divide and conquer strategy
- **Nodes** are tests for feature values
  - There is one branch for every value that the feature can take
- **Leaves** of the tree specify the class labels

- Given a collection of examples, **learn a decision tree that represents it.**
- Use this representation to **classify new examples**
  - The tree can be used for non-parametric classification and regression

Decision Tree Representation





Expressivity of Decision Trees

- What kind of Boolean functions can DT represent?
  - Any Boolean function! (***why?***)
- A decision tree can be rewritten as a DNF
  - *Each path from root to leaf can be written as a rule, the tree is a disjunction of these rules.*

  - **`Green ^ square → positive`**
  - **`Blue ^ circle → positive`**
  - **`Blue ^ square → positive`**

- **Question**: *What is the size of the hypothesis space for the shape classification problem?*

Decision Trees - representation

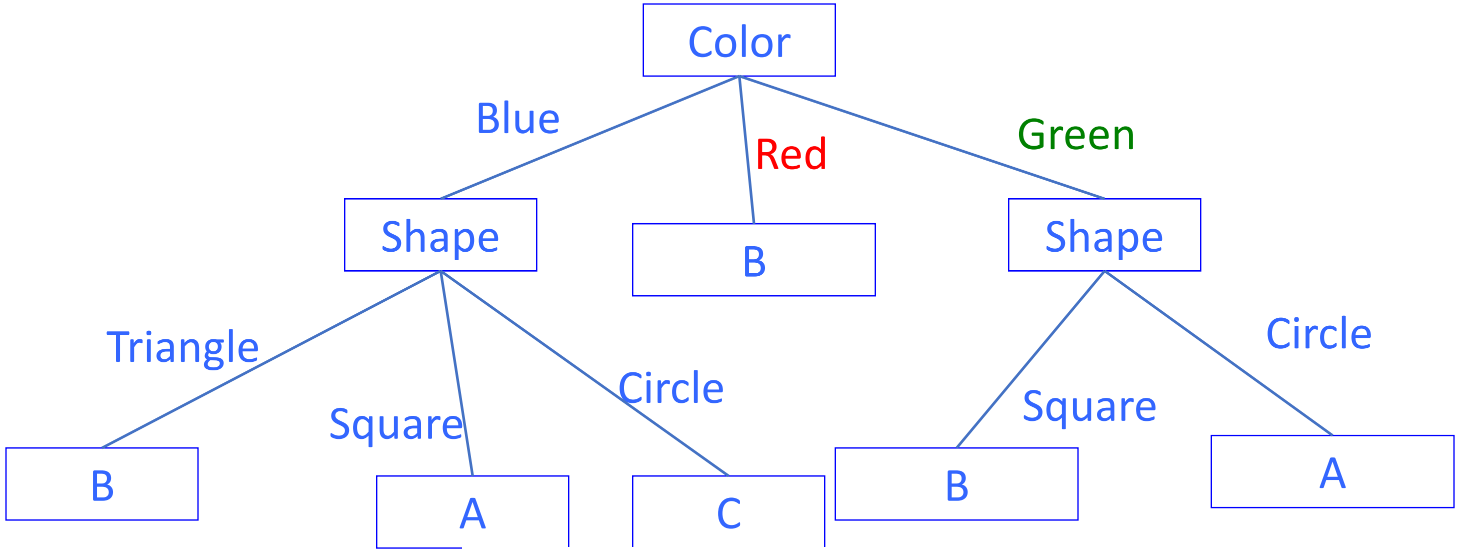
- KNN maintains the training data directly.
- DT: each path is a conjunction of attributes values leading to label
  - If paths do not share information – just store the data
  - DT: paths share prefixes

Is that helpful?

- DT learning algorithm: compressed representation of the data
  - ”lossy” vs “lossless”

Decision Tree Representation

- **Basic Questions:**
  - *How to use Decision Trees for prediction?*
    - **follow the path from the root**
      - What is the label of *a red triangle*? *Green triangle*?
  - How can we learn decision trees from data?



Decision Trees - representation

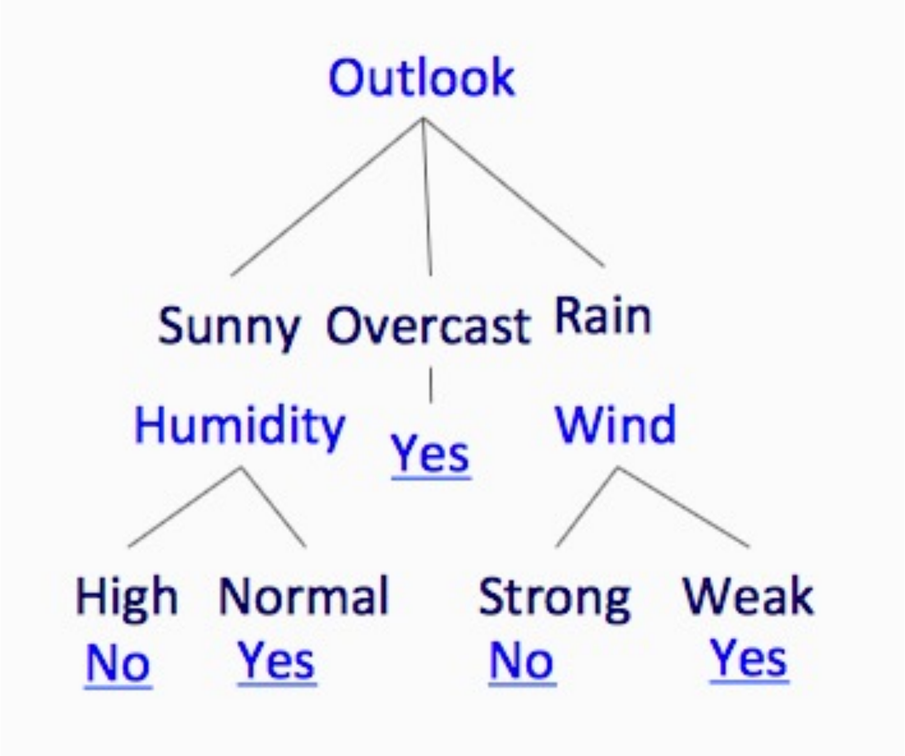
- Let’s compare **M-out-N rule**s, **KNN** and **DT**
- Assume we have a text classification problem, sentiment analysis.

- How would you represent the input document?
- What kind of dependencies will be represented by each function space?
- What would be the expected behavior on real-data? What can go “right” or “wrong”?

Basic Decision Tree Learning Algorithm

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

- Data is processed in Batch (i.e. all the data available)
- Recursively build a DT top down.



Basic Decision Tree Learning Algorithm: ID3

Input:
S the set of Examples
Label is the target attribute (the prediction)
Attributes: set of measured attributes

- 1. **If** all examples are have same label:
  - *Return a single node tree with the label*
- 2. **Otherwise**
  - Create a Root node for tree
  - A = attribute in Attributes that ***best*** classifies S
  - for each possible value v of attribute A:
    - Add a new tree branch corresponding to A=v
    - Let *Sv* be the subset of examples in S with A=v
    - **if** *Sv* is empty:

      add leaf node with the common value of Label in S

      Why?
    - **Else:**

      below this branch add the subtree:

      ID3(*Sv*, Attributes - {a}, Label)

      Recursive call
- 4. **Return** Root node

Decision Trees - representation

- **Recall our text classification example…**
- *Assume we have a text classification problem, sentiment analysis.*

- What words will appear at the top of the tree? Near the bottom?

- ”I like carrots but not lettuce” vs. “I like lettuce but not carrots”
- Would DT solve this problem?

Entropy

Entropy (impurity, disorder) of a set of examples S with respect to binary classification is

$$\boxed{Entropy(S) = H(S) = -p_+ \log(p_+) - p_- \log(p_-)}$$

- *The proportion of positive examples is p_+*
- *The proportion of negative examples is p_-*

In general, for a discrete probability distribution with K possible values, with probabilities {p_1, p_2, , p_K} the entropy is given by

$$\boxed{H(\{p_1, p_2, \cdots, p_K\}) = -\sum_{i=1}^{K} p_i \log(p_i)}$$

Information Gain

The *information gain* of an attribute A is the *expected reduction in entropy* caused by partitioning on this attribute

$$Gain(S, A) = Entropy(S) - \sum_{v \in \mathrm{Values}(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

S_v: the subset of examples where the value of attribute A is set to value v

Entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set

- partition of low entropy (imbalanced splits) leads to high gain

Go back to check which of the A, B splits is better!

Will I play tennis today?

1	**O**	**T**	**H**	**W**	**Play?**
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Will I play tennis today?

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Current entropy:

$p = 9/14$

$n = 5/14$

$H(Y) =$

$-(9/14)\log_2(9/14) \quad -(5/14)\log_2(5/14)$

$= 0.94$

Information Gain: outlook

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Outlook = sunny: 5 of 14 examples

$p = 2/5$ $n = 3/5$ $\mathbf{H_S = 0.971}$

Outlook = overcast: 4 of 14 examples

$p = 4/4$ $n = 0$ $\mathbf{H_o = 0}$

Outlook = rainy: 5 of 14 examples

$p = 3/5$ $n = 2/5$ $\mathbf{H_R = 0.971}$

Expected entropy:

$(5/14)\times0.971 + (4/14)\times0$
$+ (5/14)\times0.971 = \mathbf{0.694}$

Information gain:

$0.940 - 0.694 = \mathbf{0.246}$

Information Gain: outlook

1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Humidity = high:

$p = 3/7 \quad n = 4/7 \quad \mathbf{H_h = 0.985}$

Humidity = Normal:

$p = 6/7 \quad n = 1/7 \quad \mathbf{H_o = 0.592}$

Expected entropy:

$(7/14)\times 0.985 + (7/14)\times 0.592 = \mathbf{0.7885}$

Information gain:

$0.940 - 0.7885 = \mathbf{0.1515}$

Which feature to split on?

1	**O**	**T**	**H**	**W**	**Play?**
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Information gain:

Outlook: 0.246

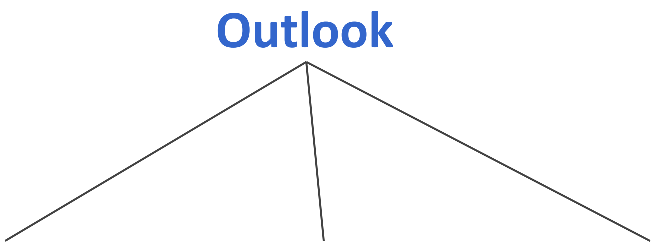
Humidity: 0.151

Wind: 0.048

Temperature: 0.029

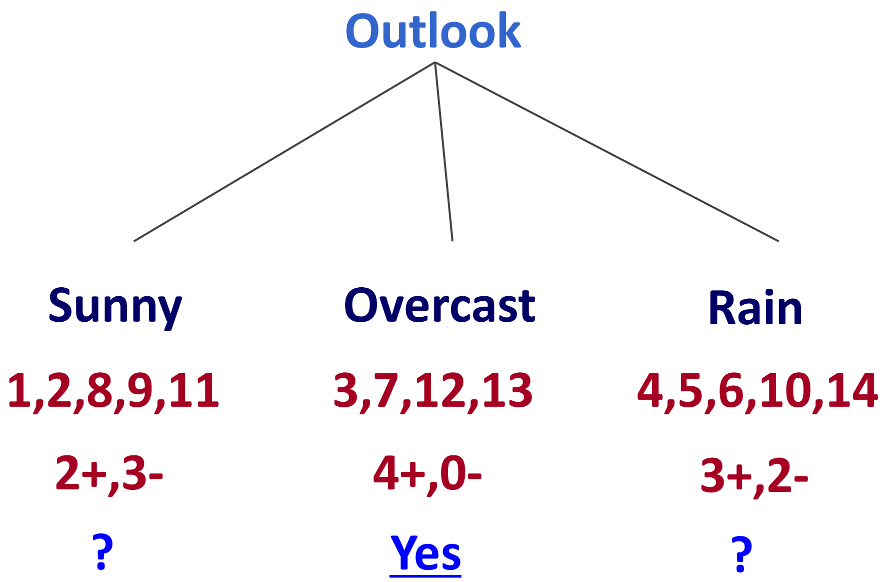
→ ***Split on Outlook***

An Illustrative Example



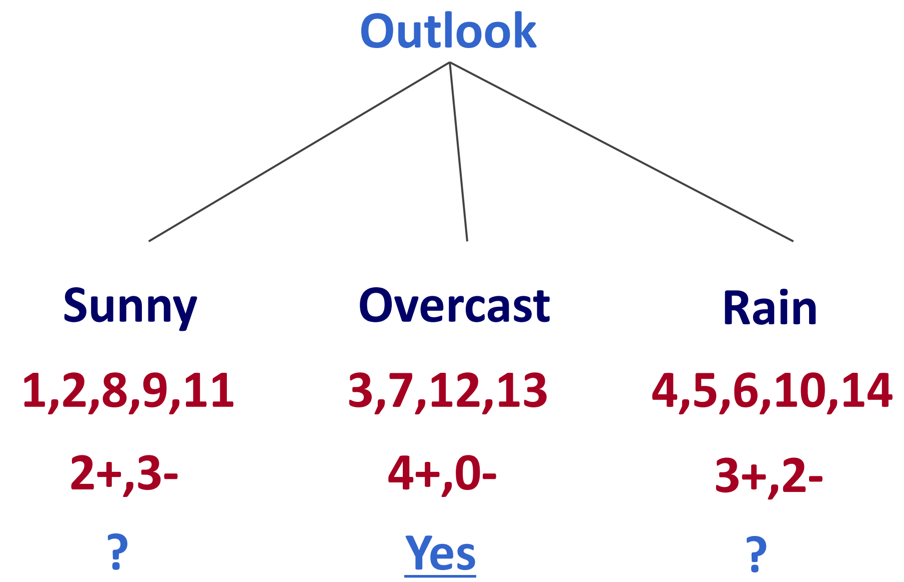
Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An Illustrative Example



1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

An Illustrative Example



1	O	T	H	W	Play?
	S	H	H	W	--
2	S	H	H	S	--
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	--
7	O	C	N	S	+
8	S	M	H	W	--
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	--

Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

An Illustrative Example

Gain(S_{sunny}, Humidity) .97-(3/5) 0-(2/5) 0= .97

Gain(S_{sunny},Temp) = .97- 0-(2/5) 1 = .57

Gain(S_{sunny}, wind) = .97-(2/5) 1 - (3/5) .92 = .02

Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

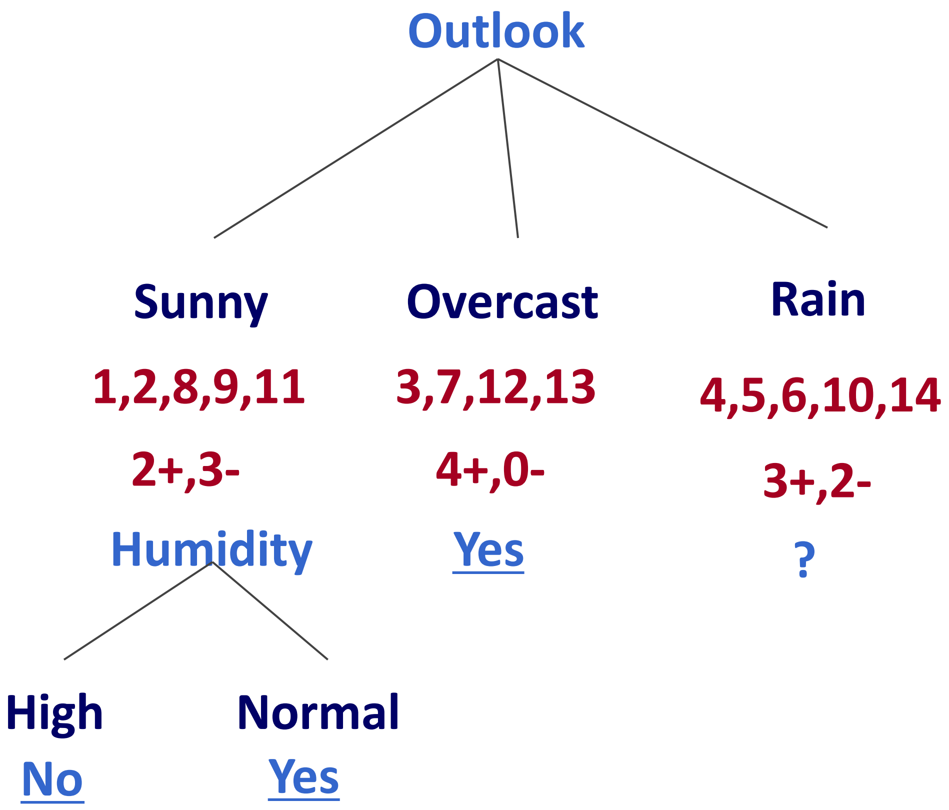
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

An Illustrative Example

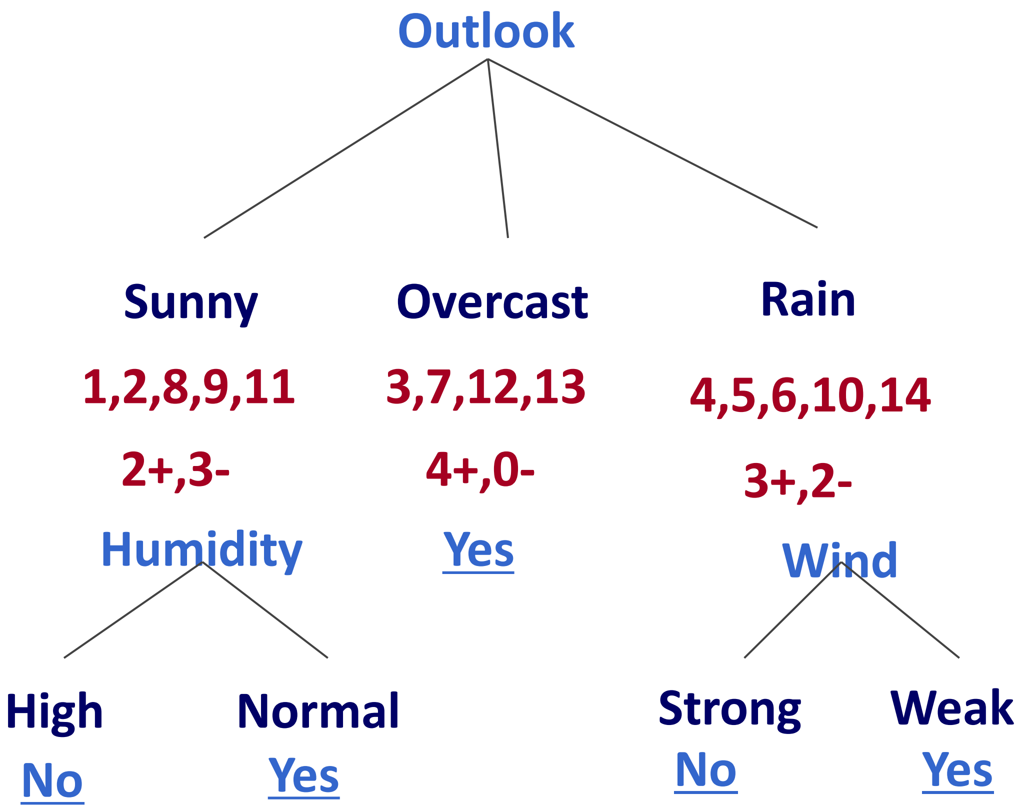
Outlook

Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

An Illustrative Example



An Illustrative Example



Variants of Information Gain

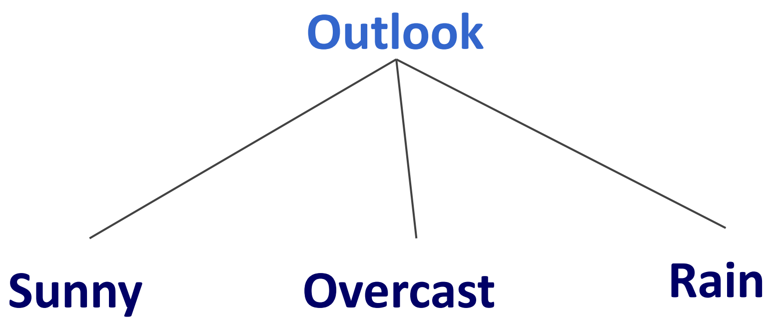
- Information gain is defined using entropy to measure the disorder/ impurity of the labels.

- **Other ways to measure disorde**r, e.g., *MajorityError*, which computes:
  - *“Suppose the tree was not grown below this node and the majority label were chosen, what would be the error?”*
  - Suppose at some node, there are 15 **+** and 5 **–** examples.
  - What is the *MajorityError*?
- Answer: ¼

- **Similar idea to entropy**

Non Boolean Features

- If the features can take multiple values
  - We have seen one edge per value



Non Boolean Features

- If the features can take multiple values
  - We have seen one edge per value (i.e a multiway split)
  - Alternative: *make the attributes Boolean by testing for each value*

Outlook=Sunny . . . [Outlook:Sunny=True,
Outlook:Overcast=False,
Outlook:Rain=False]

Continuous Attributes

What can you do with numeric features?

Use threshold *or* ranges to get Boolean tests

How should you determine the thresholds?

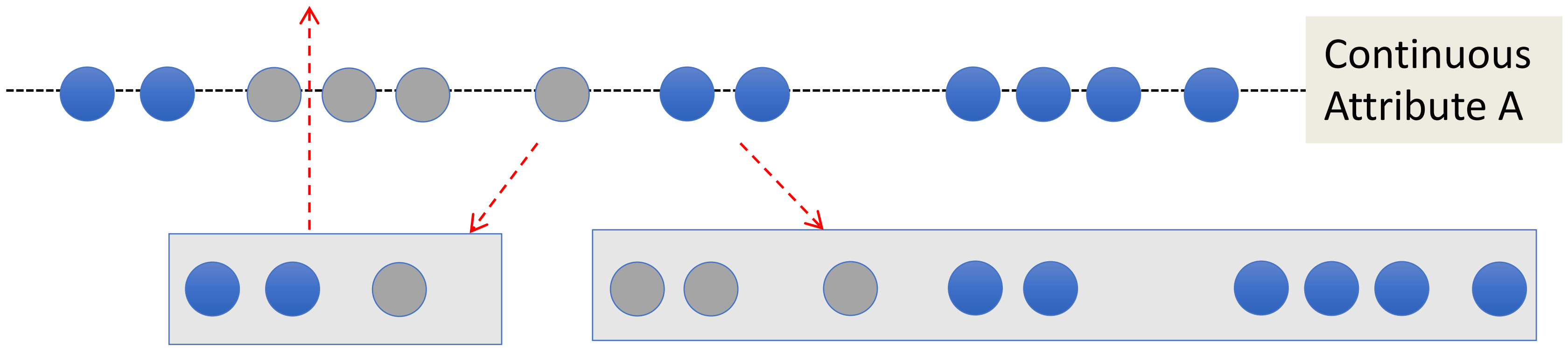
Problem:

You should consider all split points (c) to define node test $X_j > c$

Is there an easier way?

Continuous Attributes

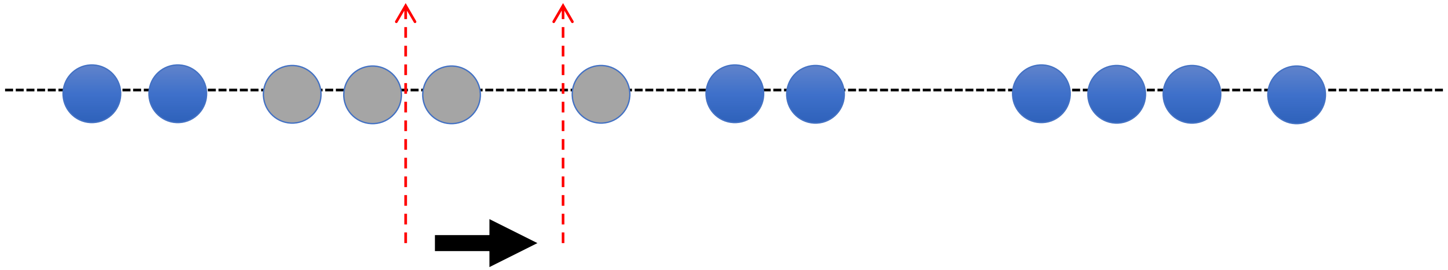
Information gain is minimized when children maintain the same distribution over output labels as the parent node



➔ The split should change the proportion of labels in the children
  ➔ Each child should have a higher proportion one of the label (different)

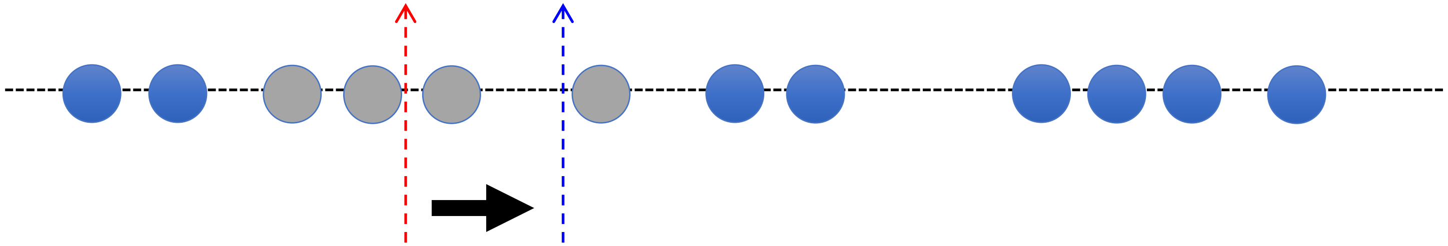
Continuous Attributes

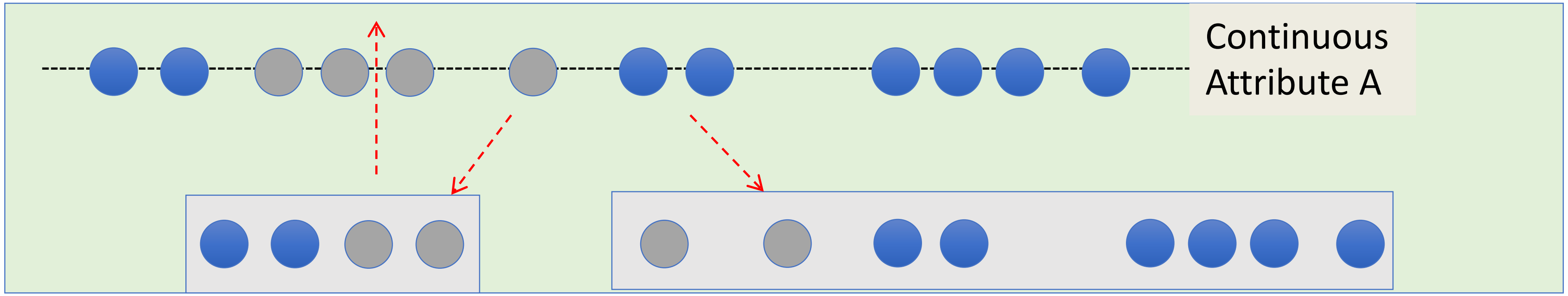
If a threshold splits two examples of the same label (say, positive), and examples on one side of the threshold (say, left), have a higher proportion than the parent distribution then:

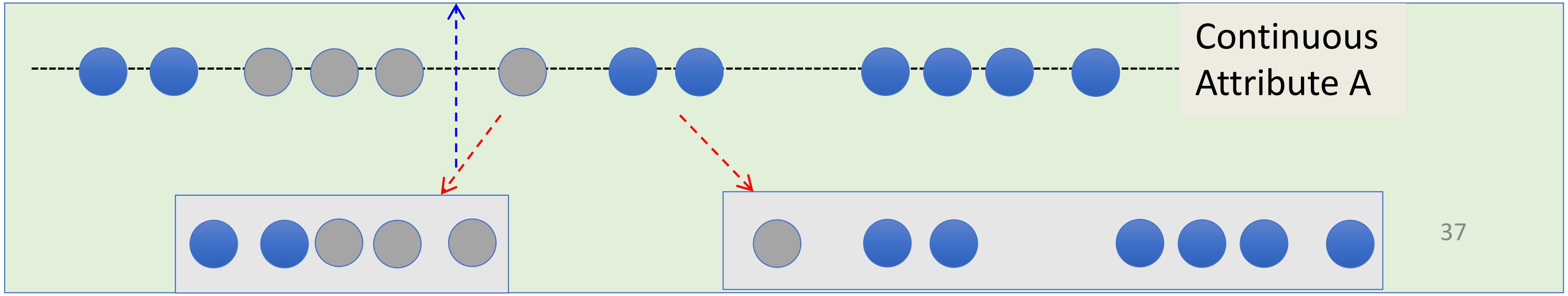


- *Examples on the other side (right) will have a higher proportion of examples with the other label (negative)*
- *Moving the threshold in this direction (right) until we get to an example with different label, will keep increasing the proportion of that label (positive/negative) in the respective children*

Continuous Attributes







Continuous Attributes

- *The highest information gain split is between examples with different labels.*
    - Simple approach: go over such split points, and find the one with highest information gain

- **Question:**

How many splits should you consider for each continuous attribute?

Bias in decision trees

- Conduct a search of the space of decision trees
  - Can represent all possible functions
- **We prefer short trees!**
  - In DT learning this is implemented as a search bias
  - We bias the search to prefer shorter trees
- Other alternatives?
  - How would you implement language bias on DT?
- Search bias is implemented using greedy heuristics
  - Hill-climbing without backtracking
  - **Overfitting can still be an issue..**

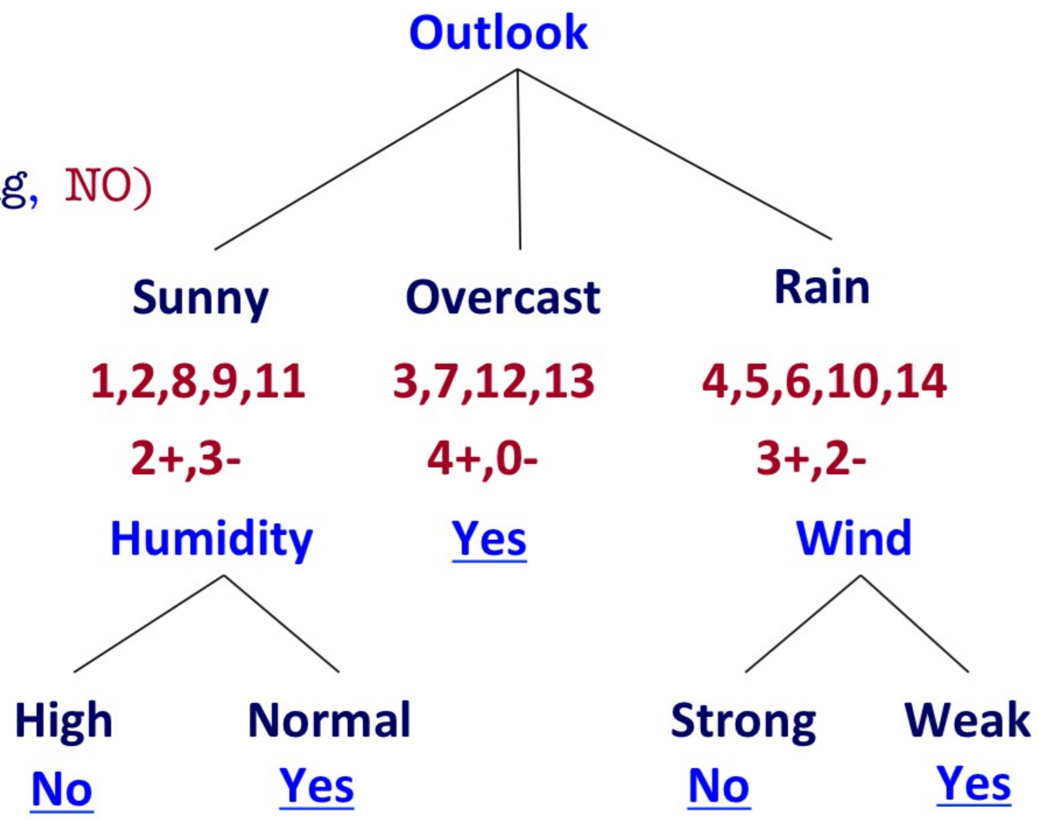
Overfitting

- Learning a tree classifying the training data perfectly may not have the best generalization
  - Algorithm fits tree to noise in training data
  - Sparse data set

A hypothesis h is said to overfit the training data if there is another hypothesis h’, such that h has a smaller error than h’ on the training data but h has larger error on the test data.

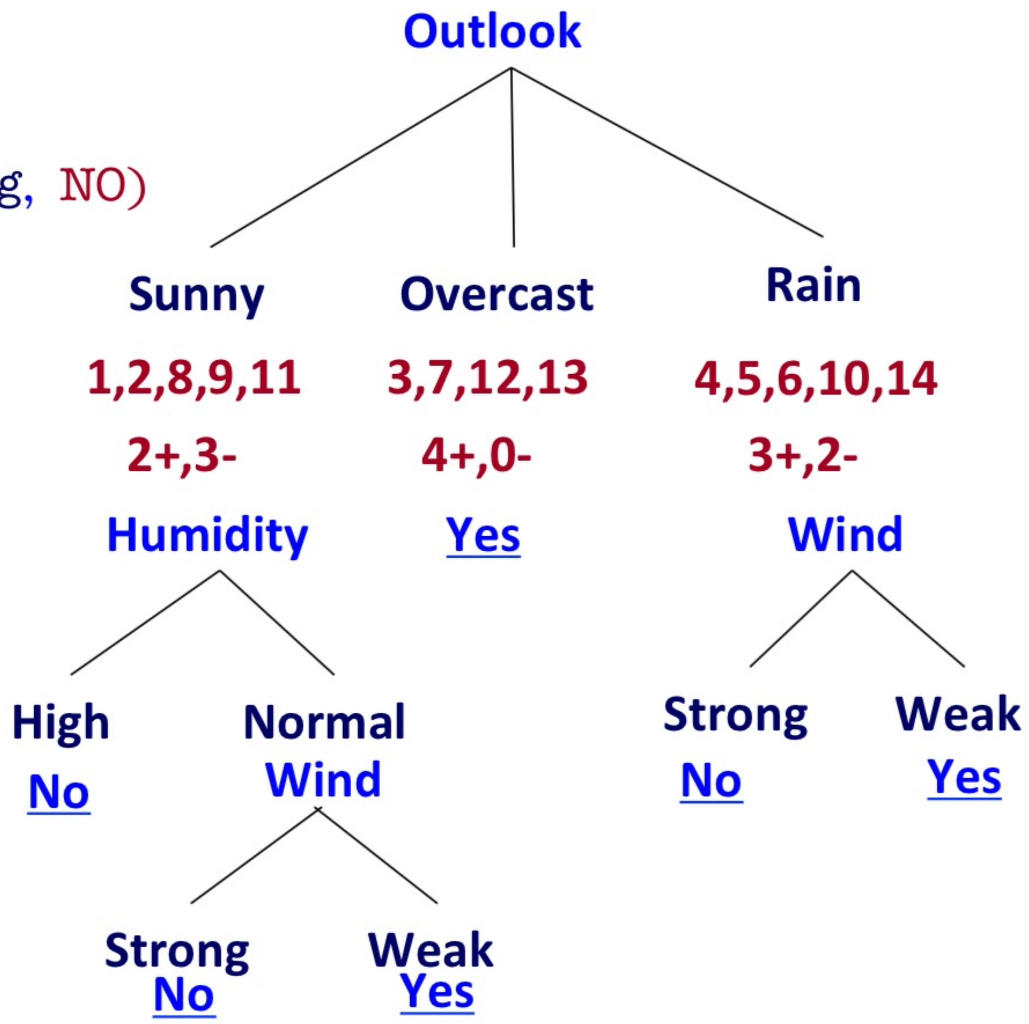
- Noisy example:

(Outlook = Sunny, Temp = Hot,
Humidity = Normal, Wind = Strong, NO)



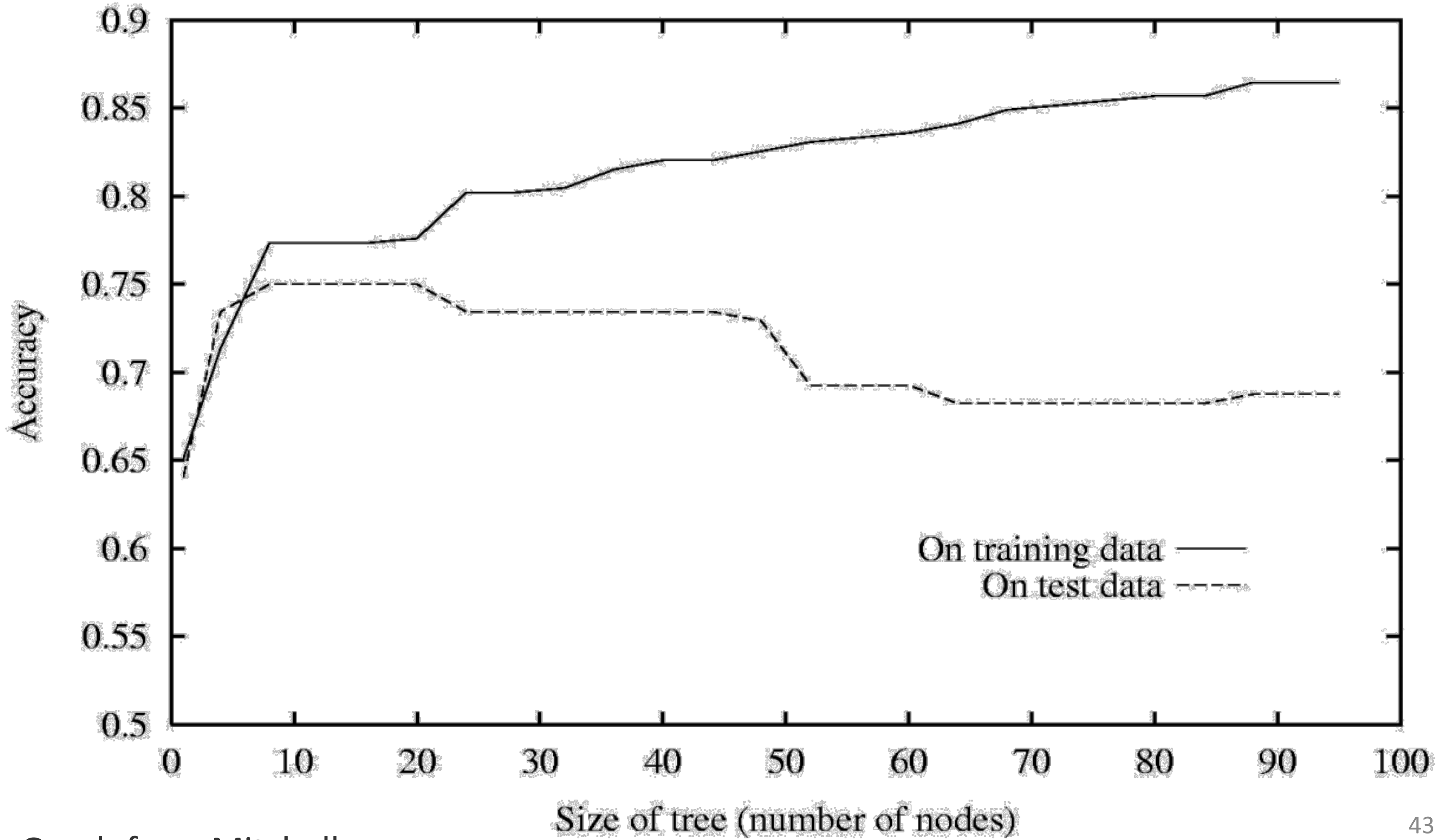
- Noisy example:

(Outlook = Sunny, Temp = Hot,
Humidity = Normal, Wind = Strong, NO)



may fit noise or other coincidental regularities

Decision trees *will* overfit



Graph from Mitchell

Avoiding overfitting in decision trees

- **Occam's Razor**
  - Favor simpler (in this case, shorter) hypotheses
  - Fewer shorter trees, less likely to fit better by coincidence
- **Static**: Fix the depth of the tree
  - Only allow trees of size K
    - Tune K using held-out validation set
    - ***Decision stump*** = a decision tree with only one level
- **Dynamic**: optimize while growing the tree
  - Grow tree on training data
  - Check performance on held-out data after adding a new node

Avoiding overfitting in decision trees

- **Occam's Razor**
  - Favor simpler (in this case, shorter) hypotheses
  - Fewer shorter trees, less likely to fit better by coincidence

- **Post Pruning:**
- While accuracy on validation set decreases. Bottom up:
  - For each non leaf node:
    - Replace sub-tree under node by a majority vote
    - Test accuracy on validation set

Decision Trees as Features

- When learning over a large number of features, learning decision trees is difficult and the resulting tree may be very large
- **Instead of pruning you can try:**
  - learn small decision trees, with limited depth.
  - Then, learn another function over these trees

- For example, Linear combination of decision stumps

Summary

- **Very popular tool**
  - Prediction is easy (and cheap!)
  - *Expressive and easy to interpret*
  - "debugging" the model is easy!
- **Learning**: greedy heuristic for representing the data
  - ID3 based on information gain
- Prone to **overfitting**!
  - Several ways to deal with it!

Further reading

Machine Learning. Tom Mitchell.

Chapter 3

A Course in Machine Learning. Hal Daumé III.

Chapter 1

(available on line: http://ciml.info/dl/v0_9/ciml-v0_9-ch01.pdf)

Questions

- What is *inductive bias*? Why is it important?
- What is the difference between **Language** bias and **search** bias?
- *Which hypothesis space is more expressive?*
  - Boolean functions , Decision trees, linear functions
- How does tree size effect generalization?
- What is the main decision when learning DT?
- How can you deal with noise? Missing attributes? Continuous values?
- *why are decision trees popular?*
  - *When should you use them?*
- *Do you think you can implement a decision tree?*

model evaluation

Model Selection

- All the algorithms that we saw (and that we will see) can be parameterized, to help control their behavior
  - *I.e., control the properties of the type of models they will produce.*
- These can capture preferences that we have about the classifiers
  - *Smaller trees, preference towards one type of error, etc.*
- In some cases, we just want the settings that get us the **best** classifier
  - *But, well.. what is **best**? and how do we know that we found it? (what can we trust?)*

Model Selection

- *We can think about selecting the best model as a* <u>secondary learning problem</u>
  - Split the data into: (1) **train** set (2) **test** set
  - Split the **train** data into: (1) **train** set (2) **validation** set
  - **<u>Training</u>**: train m models, with different parameters
    - E.g., Different ways to control the size of the tree
  - **<u>Validation</u>:** estimate the prediction error for each model
  - **<u>Testing</u>**: use the model with the least validation error
- ***The secondary learning problem***:
  - New hypothesis space: m different hypothesis to chose from
  - Pick the one that minimizes validation error

Model Selection

- ***What are the algorithm hyper-parameters?***
  - ***Decision trees:***
    - Depth of the tree
    - Pruning strategy
    - Pruning decision
    - Attribute selection heuristic
    - *Other choices?*
  - ***KNN***
    - Value of K
    - Similarity metric
- ***Every learning algorithm we will cover has a set of hyper-parameters***

K-Fold Cross validation

- *This approach is "risky"* ***(why?)***
  - Random selection of training examples for train/test/validation
    - *You could be very unlucky*
  - *Your validation data may not reflect the same distribution as your test data*
  - Optimizing on the validation data will lead to worse performance

K-Fold Cross validation

- You could get really unlucky..
  - *..but that's not likely to happen too frequently!*

- K-Fold cross validation: repeat the process K times, and average the results.

- Randomly partition the data into K equal-size subsets $S_1..S_k$
  - For i=1... K
    - Train a hypothesis on $S_1..S_{i-1}\ S_{i+1}..S_k$
    - Evaluate on S_i ($Err(S_i)$)
  - Return ($\Sigma_i\ Err(S_i)/K$)

Evaluating the Learned Hypothesis

- What is the error of h?
  - *How do I know my classifier is good enough?*
    - For example, $Err_p(h) < 0.1$
    - Is the ***training error*** a good estimate of the error?
    - ***Testing error***?
- <u>*What is the error we are "really" after?*</u>

Learning: *a few formal definitions*

Intuitions:

- Learning algorithm gets a training set (*batch* mode)
- Performance should be measured on unseen test data
- Learning works if there is a strong relationship between the *training* and *test* data
- We can trust our evaluation if there is a strong relationship between the *test* data and the "real world"

Let's formalize these intuitions!

Loss functions

- To formalize performance let's define a loss function:

$$loss(y, \hat{y})$$

  - Where $\hat{y}$ is the gold label
- The loss function measures the error on a single instance
  - Specific definition depends on the learning task

Regression

$$loss(y, \hat{y}) = (y - \hat{y})^2$$

Binary classification

$$loss(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & otherwise \end{cases}$$

Formalizing the learning process

- We assume the data is sampled from an *unknown* distribution D = P(x,y)
  - D assigns high probability to "reasonable" (x,y) pairs
  - Unreasonable (x,y) pairs:
    - *x is an unusual input (*Purdue + Hot days + January*)*
    - *Y an unlikely label for x (*Purdue + January ➔ Swimming*)*
- Performance is defined with respect to:
  - **Loss function**: What "*matters*" in the learning task
  - **Data generating distribution** (D)

Learning Definitions

Learning : representing $P(x, y) = P(y|x)P(x)$

- *P(x) is the statistical model of the "world"*
  - Producing examples according to a distribution (unknown)

- *P(y|x) statistical model of the "teacher"*
  - Generalizing the concept of the teacher
    - Target function– very peaked distribution over y for a given x

$$P(sick|Temprature > 95) = 1$$

    - Now, think about a probabilistic process labeling the data

$$P(sick|Temprature > 95) = 0.8$$

Learning Definitions

$$P(S) = P((x_1, y_1), ..., (x_n, y_n))$$

- Given S, *a dataset*, examples in S are assumed to be *Independent and identically distributed* (iid):

$$P(S) = \prod_i P(x_i, y_i)$$

  - *Independently drawn from the same distribution*
    - When are examples not independent?
    - When are examples not identically distributed?
  - The key assumption behind machine learning algorithms and their theoretic analysis.

Formalizing the learning process

- Learning goal: Minimize expected loss over D w.r.t *l*

$$\epsilon \triangleq \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\ell(y, f(x))\right] = \sum_{(x,y)} \mathcal{D}(x, y)\ell(y, f(x))$$

We do not know D in advance!

- *But, we have access to training data sampled from D*
- *Instead, compute Empirical loss*
  - ***What is the difference?***

$$\hat{\epsilon} \triangleq \frac{1}{N}\sum_{n=1}^{N} \ell(y_n, f(x_n))$$

- **Learning**: *find a function with low **expected** loss over D w.r.t l.*
- ***This is where inductive bias comes in!***

Evaluating the Learned Hypothesis

- What is a “good” error for a learned hypothesis?
  - ***How do I know my classifier is good enough?***
    - For example, $Err_p(h) < 0.1$
    - Is the training error a good estimate of the error?
    - Testing error? *Is it a good approximation for the* ***true error****?*
      - On average, that’s the true performance.
      - We have to account to variability!
      - i.e., different choices of testing sets could lead to different results!

  - **How can we account for the test error variability ?**
    - Run multiple times, and average results
    - Closed form solution

Evaluating the Learned Hypothesis

- **How are errors generated?**
  - *S (test set) randomly draws an example, and get a label from the classifier*
  - *P(first example is wrong)?*
    - ***True error***
  - *P(second example is wrong)?*
    - ***True error***
  - Draws are independent

  This should sound really familiar!
- N “coin flips” (testing examples)
- What is the probability of K errors?

➔ ***Number of errors is binomially distributed!***

Quick Detour: *Binomial Distribution*

$$P(X = x|p, n) = \frac{n!}{x!(n-x)!}\, p^x (1-p)^{n-x}$$

x number of errors we observe
p is the probability of errors
n is the number of test examples

Evaluating the Learned Hypothesis

- Our algorithm produced a model (h)
  - 10 errors out of 500 test examples
  - Is that significant evidence for the $Err_P(h)<$**0.1?**
- Significance test:
  - **Null hypothesis**: p ≥ **0.1**
  - *what is the probability that we see at most 10 errors out of 500?*

$$P(x \leq 10 | p = 0.1, n = 500) \approx 10^{-12} \leq 0.05$$

  - *If the null hypothesis was true, the observed performance is unlikely*
  - ***We can reject the null hypothesis!***

Precision and Recall

- Given a dataset, we train a classifier that gets 99% accuracy
- **Did we do a good job?**
- Build a classifier for brain tumor:
  - 99.9% of brain scans do not show signs of tumor
  - ***Did we do a good job?***
- By simply saying “NO” to all examples we reduce the error by a factor of 10!
  - ***Clearly Accuracy is not the best way to evaluate the learning system when the data is heavily skewed!***
- **Intuition**: we need a measure that captures the class we care about! (rare)

Precision and Recall

- The learner can make two kinds of mistakes:
  - False Positive
  - False Negative

	True Label: 1	True Label: 0
Predicted: 1	True Positive	False Positive
Predicted: 0	False Negative	True Negative

- **Precision**:
- *"when we predicted the rare class, how often are we right?"*

$$\frac{\text{True Pos}}{\text{Predicted Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos}}$$

- **Recall**
- "Out of all the instances of the rare class, how many did we catch?"

$$\frac{\text{True Pos}}{\text{Actual Pos}} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Neg}}$$

F-Score

- **Precision and Recall give us two reference points to compare learning performance**

	Precision	**Recall**
Algorithm 1	0.5	0.4
Algorithm 2	0.7	0.1
Algorithm 3	0.02	1

- *Which algorithm is better?*

We need a single score

- Option 1: Average $\frac{P+R}{2}$
- Option 2: F-Score $2\frac{PR}{P+R}$

Properties of f-score:

- *Ranges between 0-1*
- *Prefers precision and recall with similar values*

Ablation Study

- Making predictions often relies on many different attributes of the input object
- Let's consider email phishing detection:
  - Baseline system: lexical features
    - *"The excellent prince of Mars wants to give you a g1ft"*
    - *Accuracy : 70%*
  - *Complex system:*
    - *Lexical features, sender, email headers, servers, images, dictionaries of suspicious terms, spelling mistakes*
    - *Accuracy: 85%*
- *What aspects are responsible for the improvement?*
  - ***Run an ablation study***

Ablation Study

- Remove one feature and train+test the model
- **Other things to check:**
  - *What is the influence of features choices on* ***precision/recall****?*
  - Similar mistakes or different mistakes?

Features	ACC
Lexical features	66
Sender	79
Email headers	83
Servers	80
Images	85
Suspicious terms	82
Baseline (lexical features)	70

Error Analysis:

You can also look at the **type** of mistakes your model is making:

Some Phishing scams could be easier to detect than others.

- *Check the influence of different features by mistake types*

Error Analysis

- Identify the root cause of the mistakes your algorithm makes
  - **Aspects not captured by your features**

"We wish you a happy new year"

  - Noisy feature extraction
    - E.g., "Suspicious terms" detector is not comprehensive enough
  - Many other reasons: noisy labels,..