CSEP 546: Decision Trees and Experimental Methodology

Jesse Davis

Outline

- Decision Trees
  - Representation
  - Learning Algorithm
  - Potential pitfalls

- Experimental Methodology

Decision Trees

- Popular hypothesis space
  - Developed with learning in mind
  - Deterministic
  - Simple learning algorithm
  - Handles noise well
  - Produce comprehensible output

Decision Trees

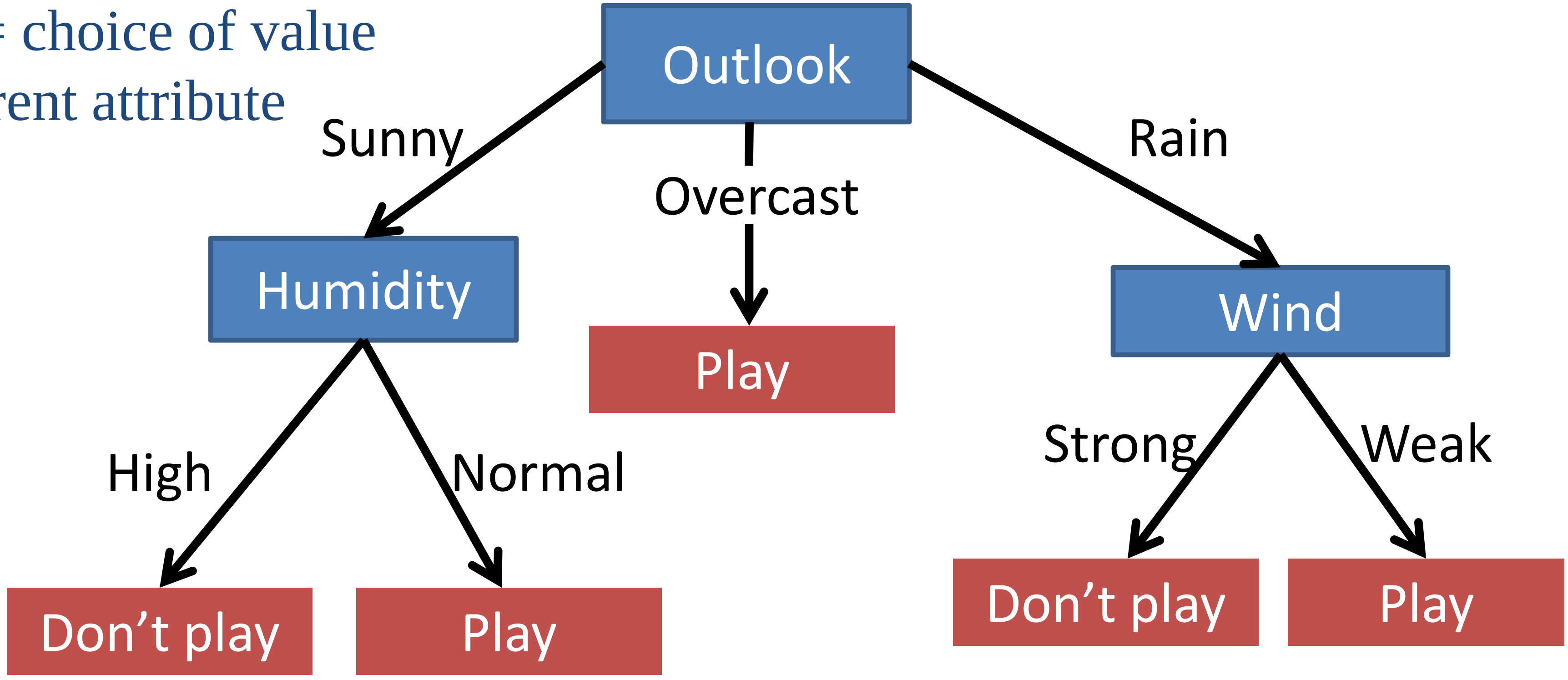
- Effective hypothesis space
  - Variable sized hypotheses
  - Can represent any Boolean function
  - Can represent both discrete and continuous features
  - Equivalent to propositional DNF
- Classify learning algorithm as follows:
  - Constructive search: Learn by adding nodes
  - Eager
  - Batch [though online algorithms exist]

Decision Tree Representation

Good day for tennis?

Leaves = classification

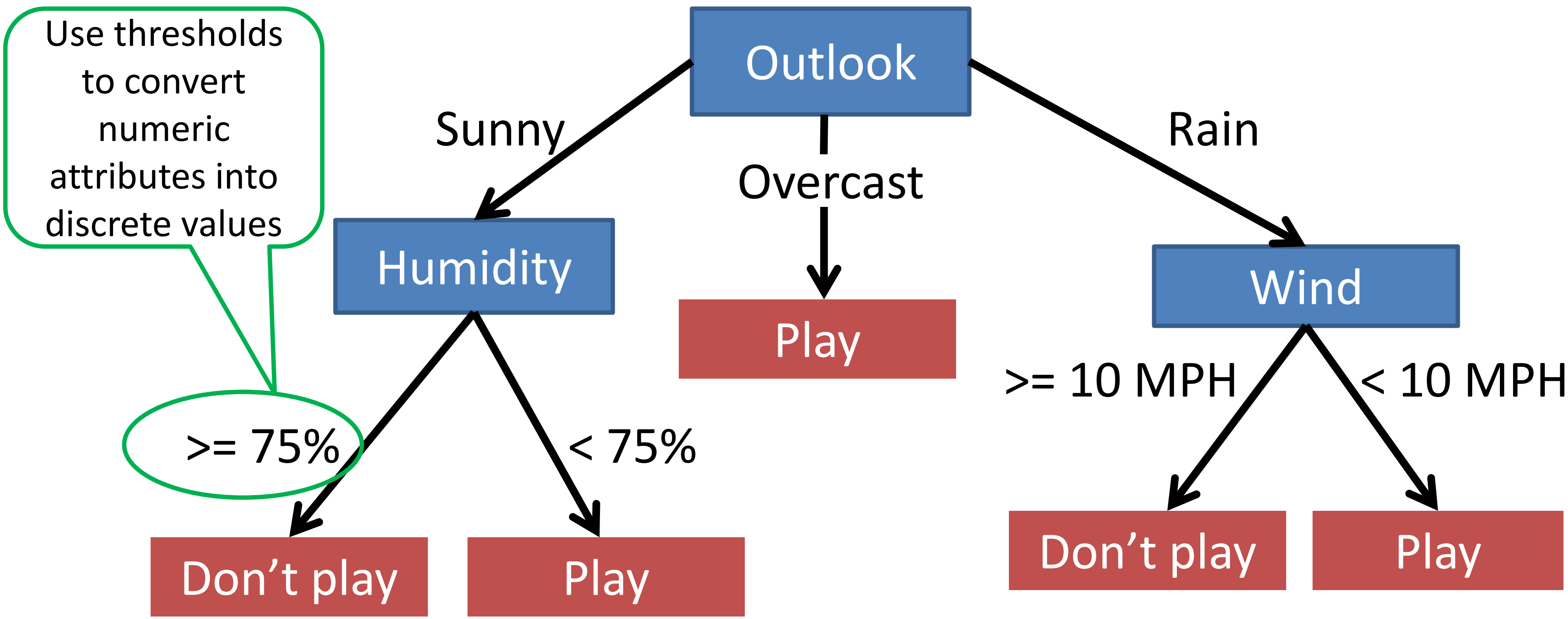
Arcs = choice of value for parent attribute



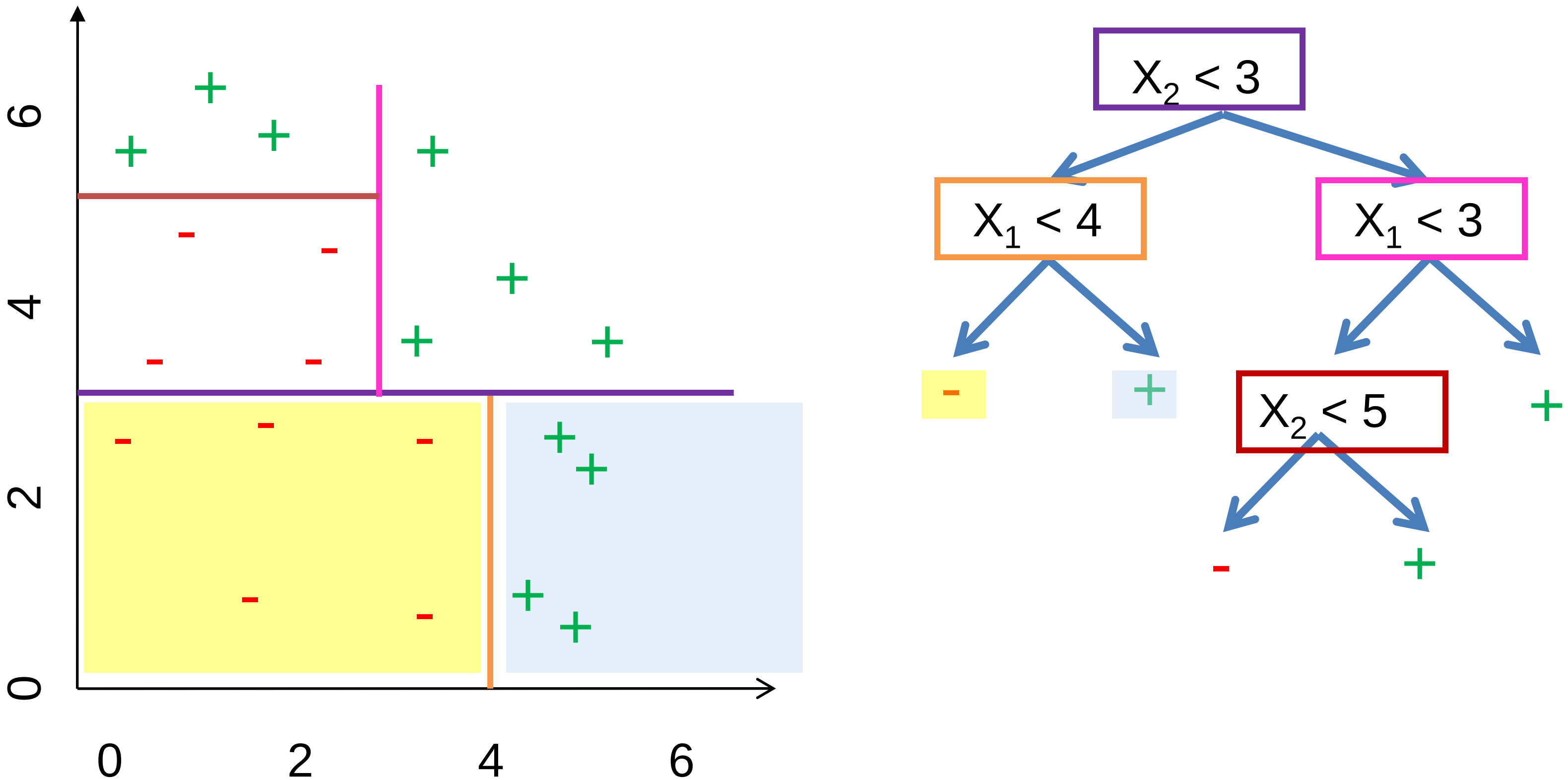
Decision tree is equivalent to logic in disjunctive normal form

Play $\Leftrightarrow$ (Sunny $\wedge$ Normal) $\vee$ Overcast $\vee$ (Rain $\wedge$ Weak)

Numeric Attributes



How Do Decision Trees Partition Feature Space?



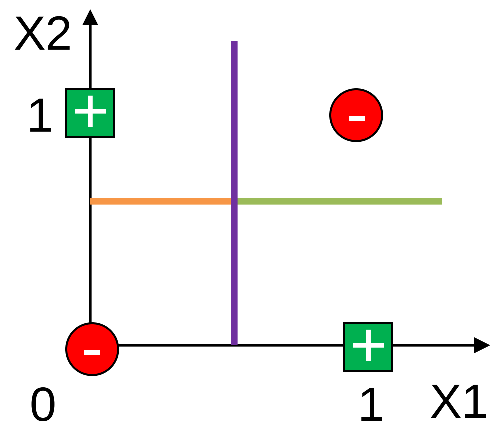
Decisions divide feature space into axis parallel rectangles and labels each one with one of the K classes

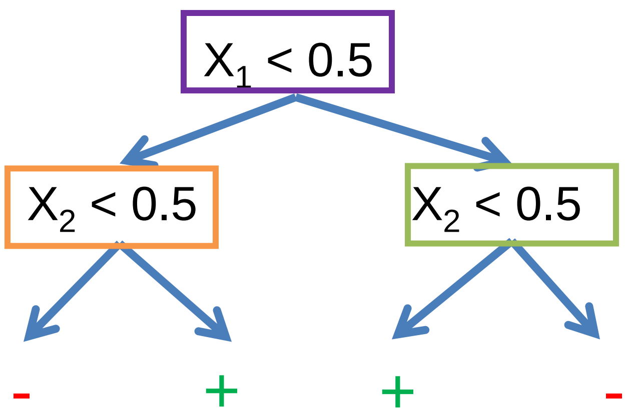
Decision Trees Provide Variable-Size Hypothesis Space

- As the number of nodes (or tree depth) increases, the hypothesis space grows
  - Depth 1 (decision "stumps"): Any Boolean function over one variable
  - Depth 2:
    - Any Boolean function over two variables
    - Some Boolean functions over three variables e.g., $(x_1 \wedge x_2) \vee (!x_1 \wedge !x_3)$
  - Etc.

Decision Trees Can Represent Any Boolean Function

	Input	Output
a)	0 0	-
b)	0 1	+
c)	1 0	+
d)	1 1	-





However, in the worst case, the tree will require exponential many nodes

Objective of DT Learning

Goal: Find the decision tree that minimizes the error rate on the training data

- Solution 1: For each training example, create one root-to-leaf path
- Problem 1: Just memorizes the training data
- Solution 2: Find smallest tree that minimizes our error function
- Problem 2: This is NP-hard
- Solution 3: Use a greedy approximation

DT Learning as Search

- Nodes

  Decision Trees:

  1) Internal: Attribute-value test

  2) Leaf: Class label

- Operators

  Tree Refinement: Sprouting the tree

- Initial node

  Smallest tree possible: a single leaf

- Heuristic?

  Information Gain

- Goal?

  Best tree possible   (???)

Decision Tree Algorithm

```
BuildTree(TraingData)
    Split(TrainingData)

Split(D)
    If (all points in D are of the same class)
        Then Return
    For each attribute A
        Evaluate splits on attribute A
    Use best split to partition D into D1, D2
    Split (D1)
    Split (D2)
```

What is the Simplest Tree?

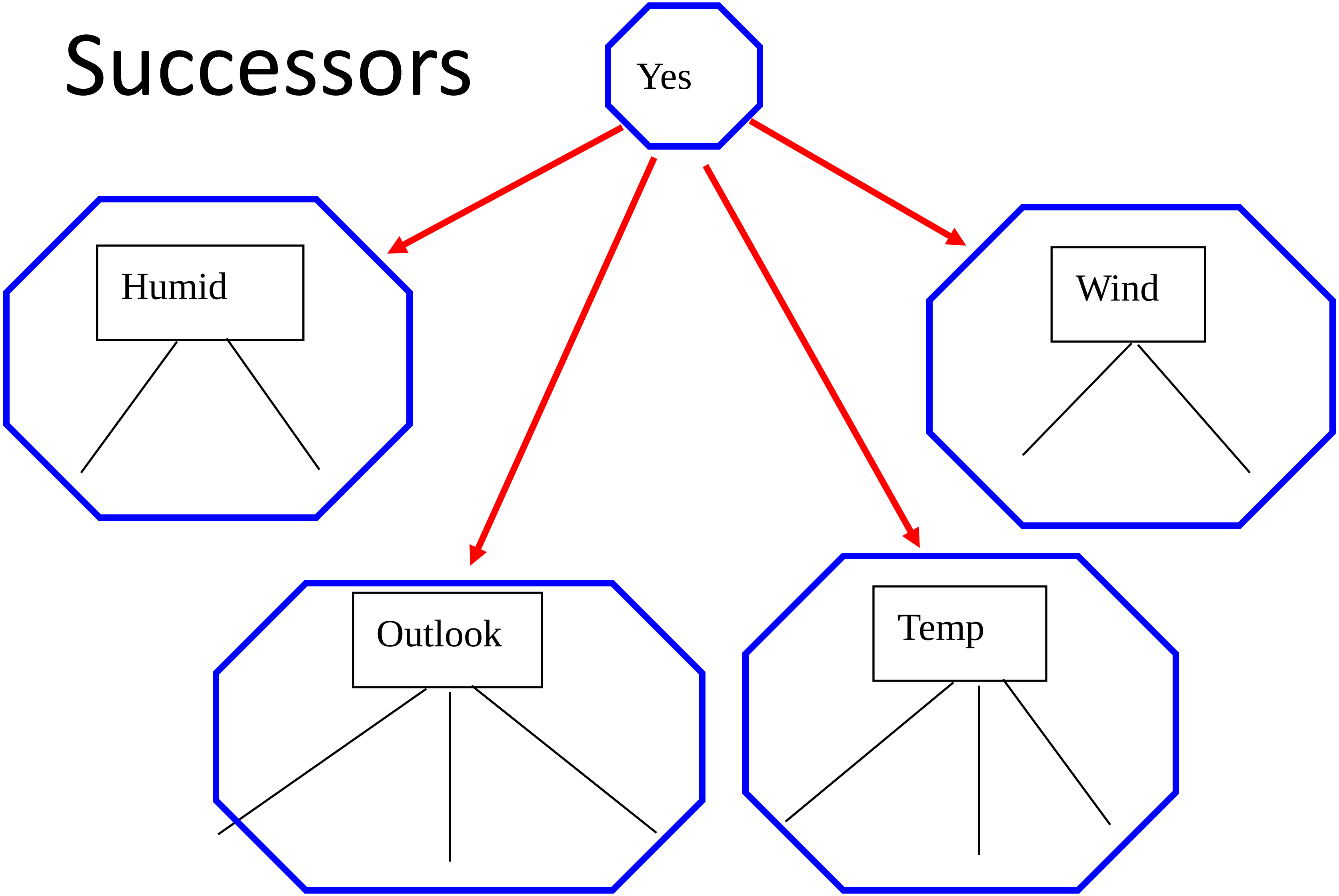
Day	Outlook	Temp	Humid	Wind	Play?
d1	s	h	h	w	n
d2	s	h	h	s	n
d3	o	h	h	w	y
d4	r	m	h	w	y
d5	r	c	n	w	y
d6	r	c	n	s	n
d7	o	c	n	s	y
d8	s	m	h	w	n
d9	s	c	n	w	y
d10	r	m	n	w	y
d11	s	m	n	s	y
d12	o	m	h	s	y
d13	o	h	n	w	y
d14	r	m	h	s	n

How good?

[9+, 5-]

Majority class:
- correct on 9 examples
- incorrect on 5 examples

Successors



Which attribute should we use to split?

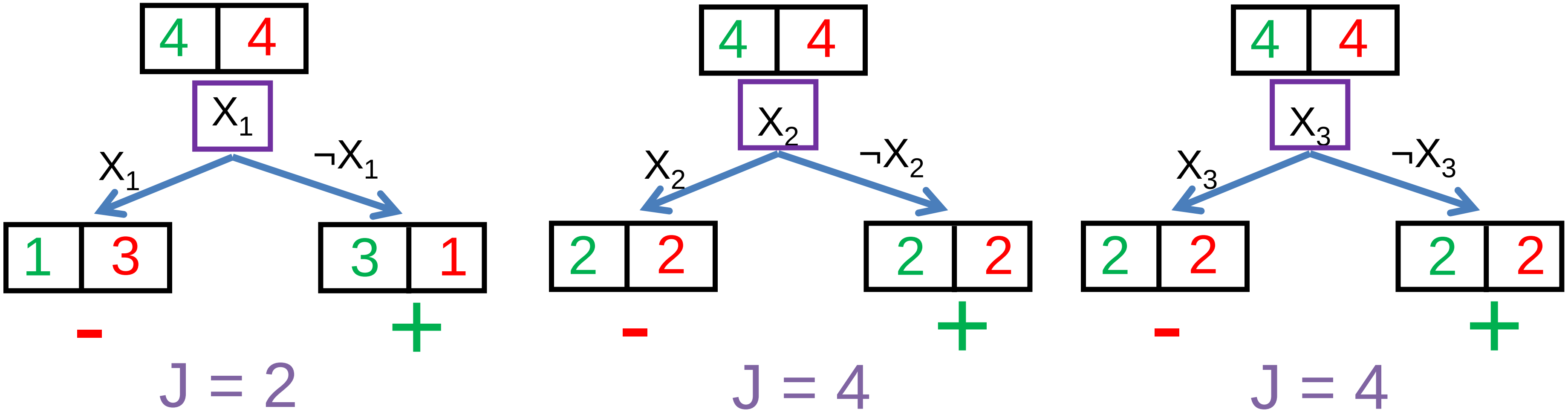
Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data.

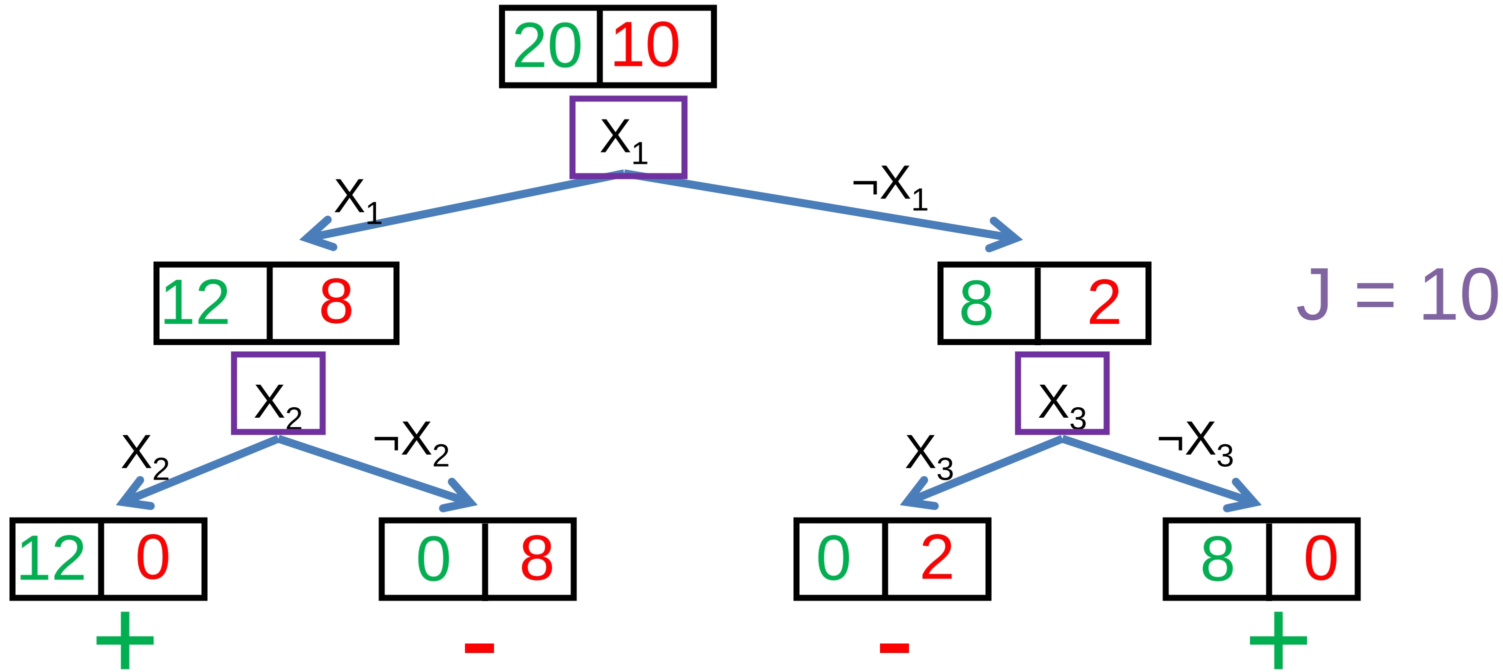
ChooseBestAttribute(S)
choose j to minimize J_j, computed as follows:
$S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
$S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
$y_0 =$ the most common value of y in S_0
$y_1 =$ the most common value of y in S_1
$J_0 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$
$J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$
$J_j = J_0 + J_1$ (total errors if we split on this feature)
return j

Choosing the Best Attribute: Example

	Input	Output
a)	0 0 0	+
b)	0 0 1	-
c)	0 1 0	+
d)	0 1 1	+
e)	1 0 0	-
f)	1 0 1	+
g)	1 1 0	-
h)	1 1 1	-

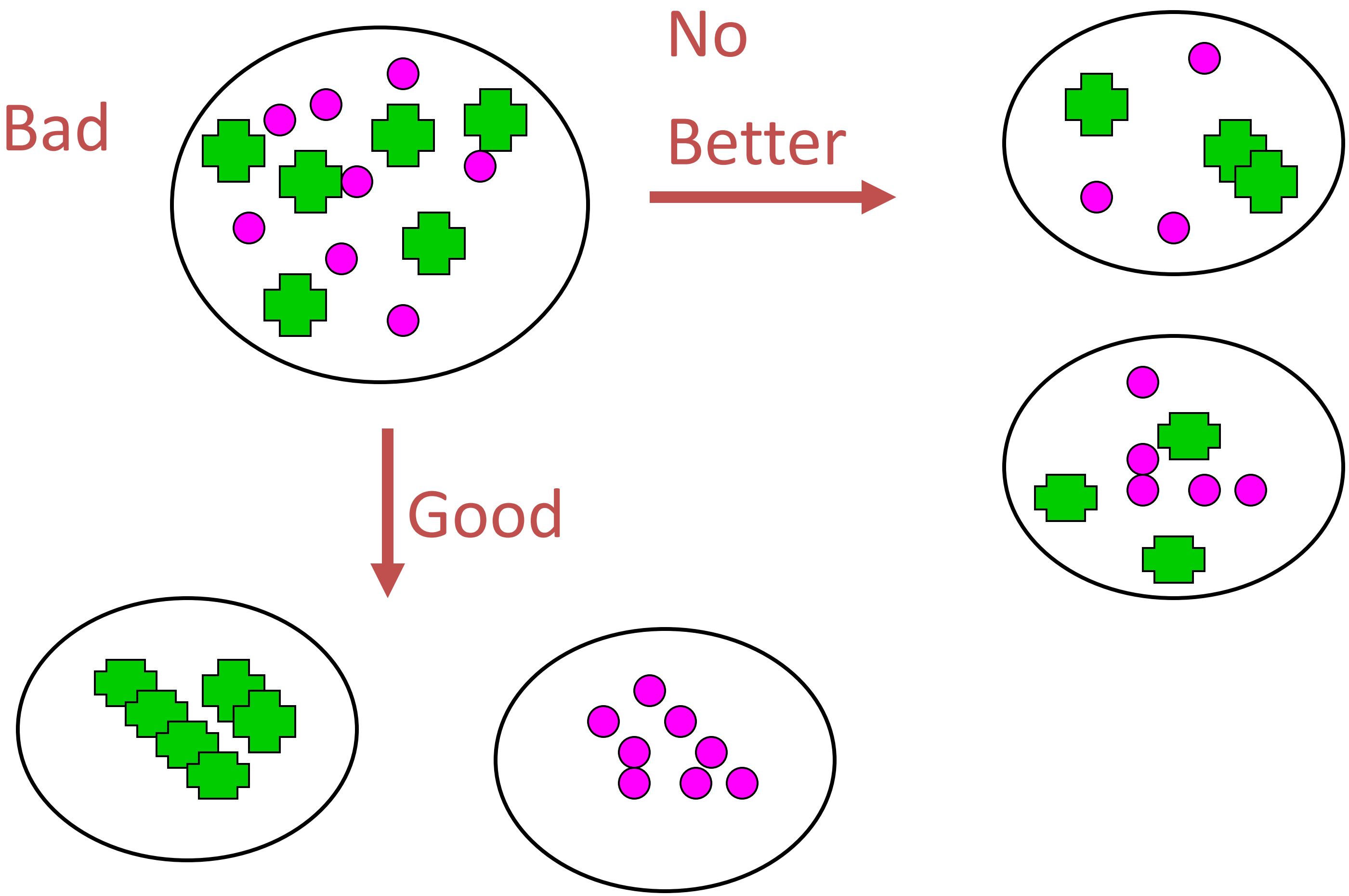


Choosing the Best Attribute: Example



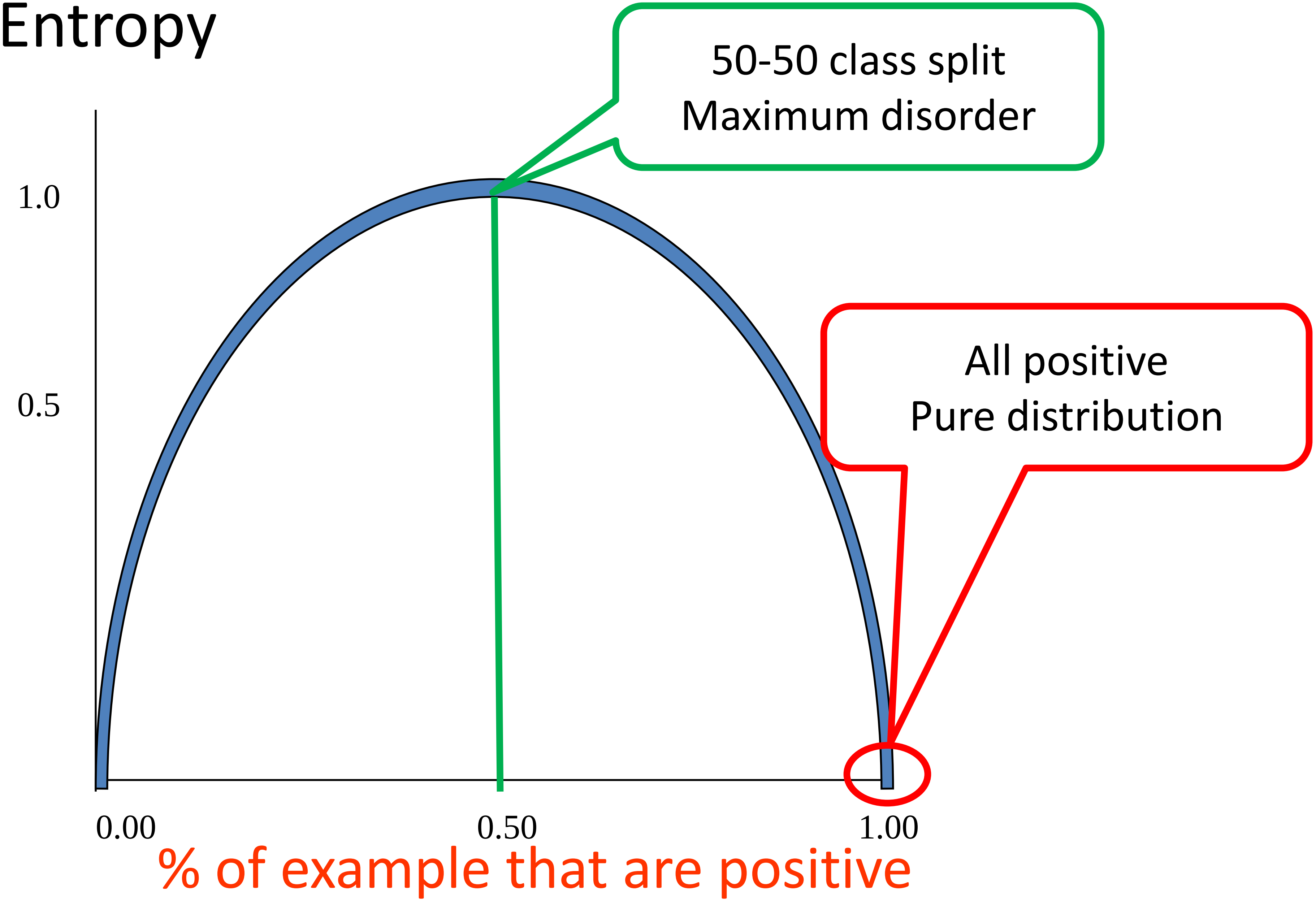
This metric may not work well as it does not always detect cases where we are making progress towards the goal

A Better Metric From Information Theory



Intuition: Disorder is bad and homogeneity is good

Entropy



50-50 class split
Maximum disorder
1.0
0.5
All positive
Pure distribution
0.00
0.50
1.00
% of example that are positive

Entropy (disorder) is bad
Homogeneity is good

- Let S be a set of examples
- Entropy(S) = $-P \log_2(P) - N \log_2(N)$
  - **P** is proportion of pos example
  - **N** is proportion of neg examples
  - $0 \log 0 == 0$
- Example: S has 9 pos and 5 neg

$$\text{Entropy}([9+, 5-]) = -(9/14) \log_2(9/14) - (5/14)\log_2(5/14)$$
$$= 0.940$$

Information Gain

- **Measure of expected *reduction* in entropy**
- **Resulting from splitting along an attribute**

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

Where $\mathrm{Entropy}(S) = -P \log_2(P) - N \log_2(N)$

Example: “Good day for tennis”

- Attributes of instances
  - Outlook = {rainy (r), overcast (o), sunny (s)}
  - Temperature = {cool (c), medium (m), hot (h)}
  - Humidity = {normal (n), high (h)}
  - Wind = {weak (w), strong (s)}
- Class value
  - Play Tennis? = {don’t play (n), play (y)}
- Feature = attribute with one value
  - E.g., outlook = ***sunny***
- Sample instance
  - outlook=***sunny,*** temp=***hot***, humidity=***high***, wind=***weak***

Experience: “Good day for tennis”

Day	Outlook	Temp	Humid	Wind	PlayTennis?
d1	s	h	h	w	n
d2	s	h	h	s	n
d3	o	h	h	w	y
d4	r	m	h	w	y
d5	r	c	n	w	y
d6	r	c	n	s	n
d7	o	c	n	s	y
d8	s	m	h	w	n
d9	s	c	n	w	y
d10	r	m	n	w	y
d11	s	m	n	s	y
d12	o	m	h	s	y
d13	o	h	n	w	y
d14	r	m	h	s	n

Gain of Splitting on Wind

Values(wind)=weak, strong

S = [9+, 5-]

S_{weak} = [6+, 2-]

S_s = [3+, 3-]

Gain(S, wind)

$$= \text{Entropy}(S) - \sum_{v \in \{weak,\, s\}} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

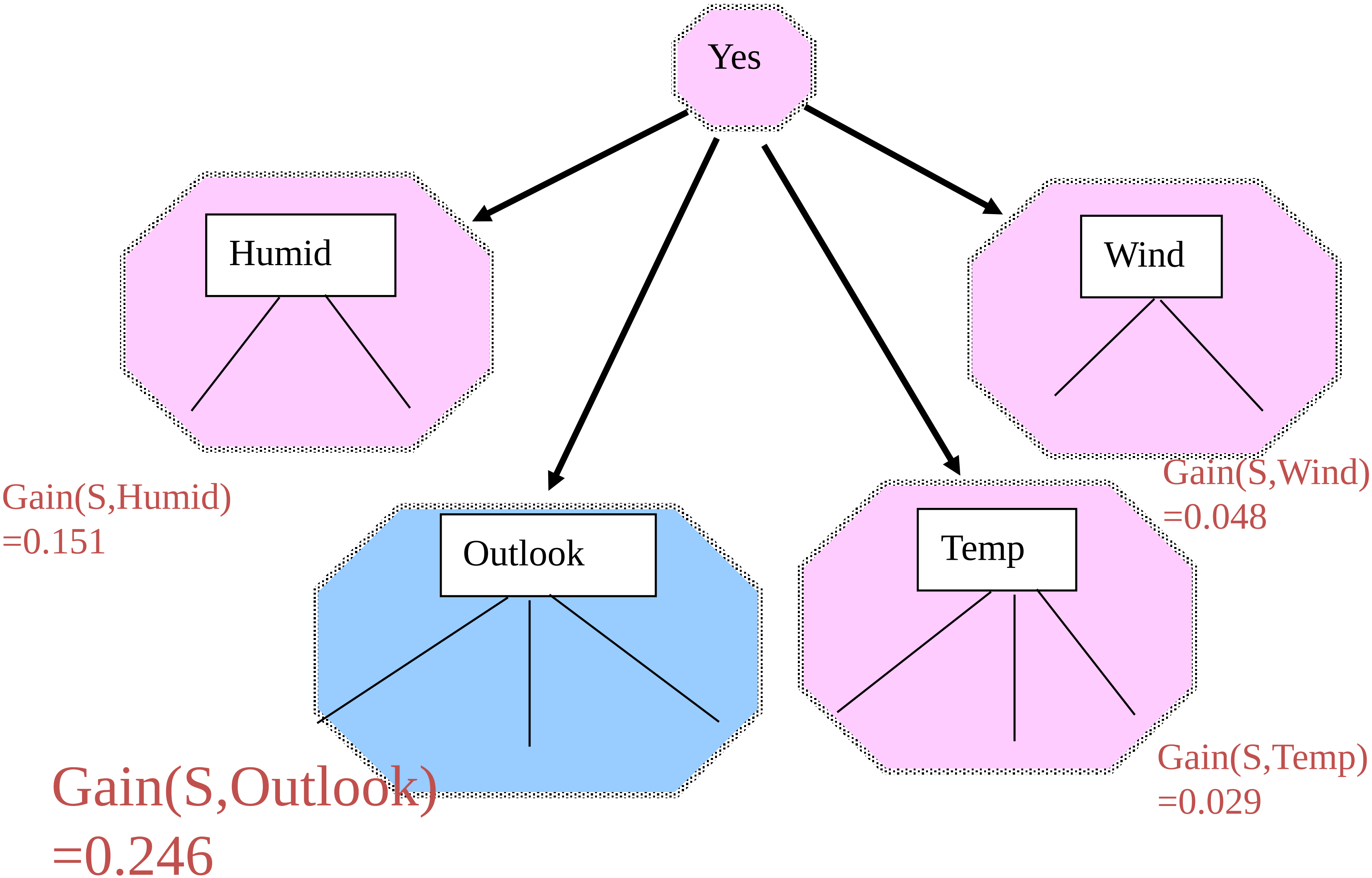
$$= \text{Entropy}(S) - 8/14\ \text{Entropy}(S_{weak}) - 6/14\ \text{Entropy}(S_s)$$

$$= 0.940 - (8/14)\ 0.811 - (6/14)\ 1.00$$

$$= .048$$

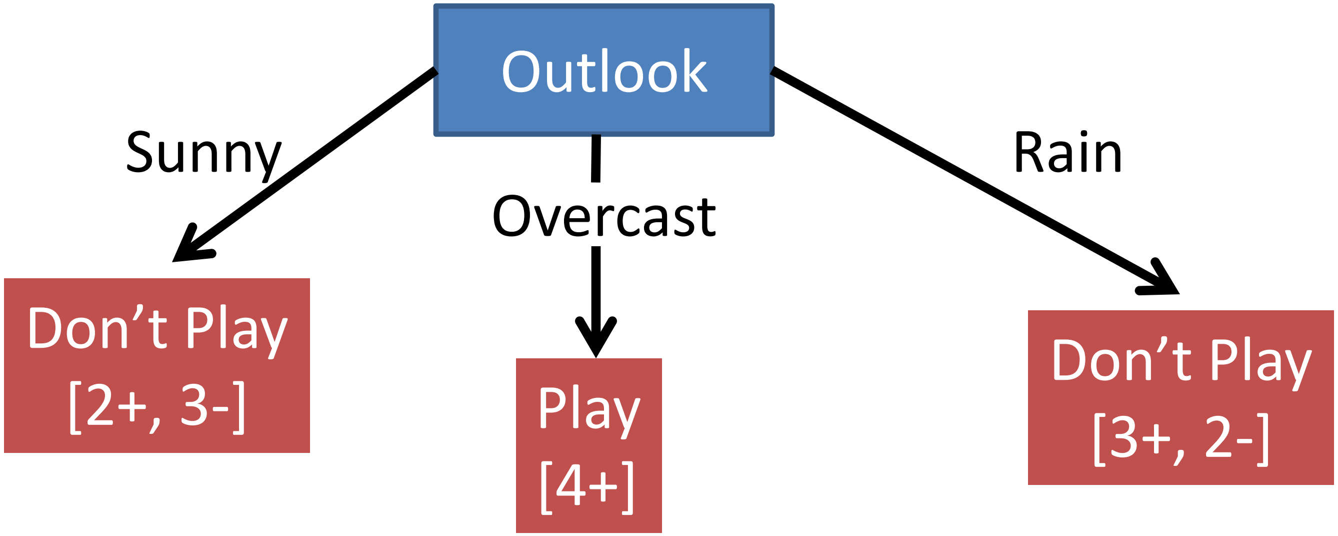
Day	Wind	Tennis?
d1	weak	n
d2	s	n
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	s	n
d7	s	yes
d8	weak	n
d9	weak	yes
d10	weak	yes
d11	s	yes
d12	s	yes
d13	weak	yes
d14	s	n

Evaluating Attributes



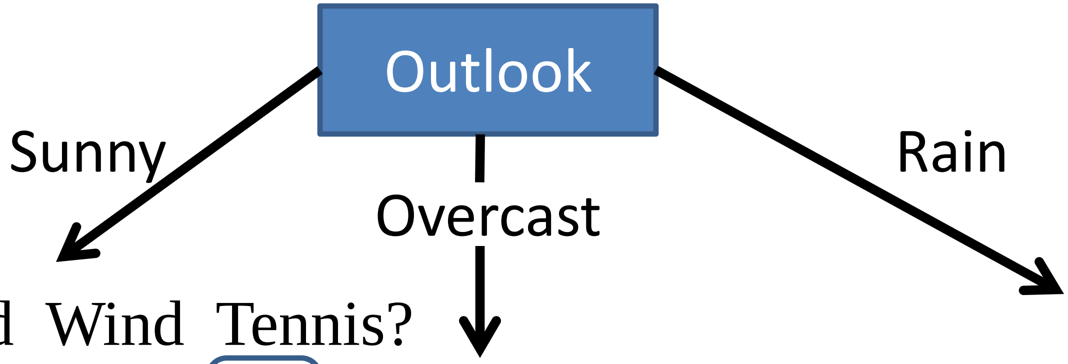
Resulting Tree

Good day for tennis?



Recurse

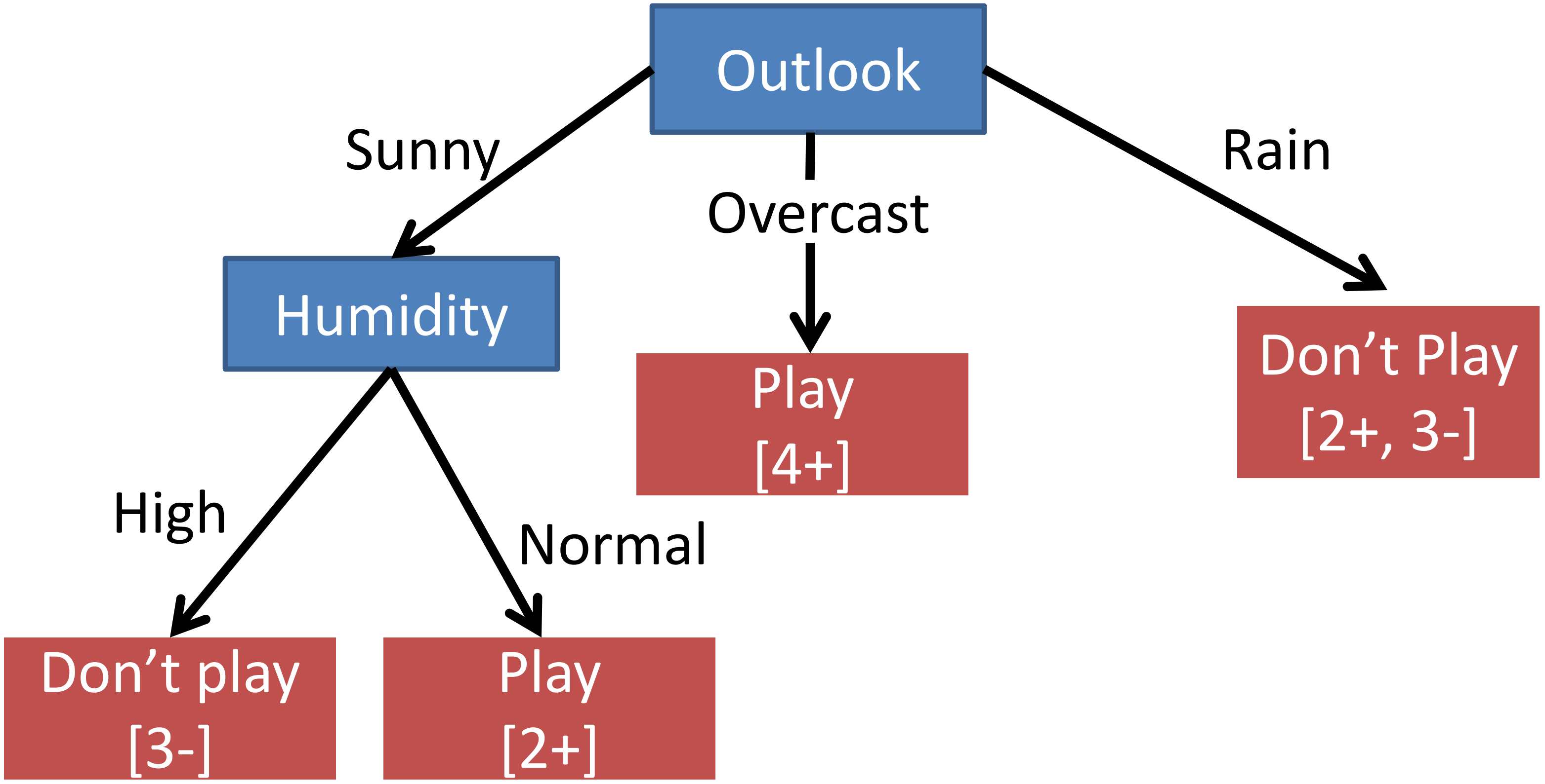
Good day for tennis?



Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	n	weak	yes
d11	m	n	s	yes

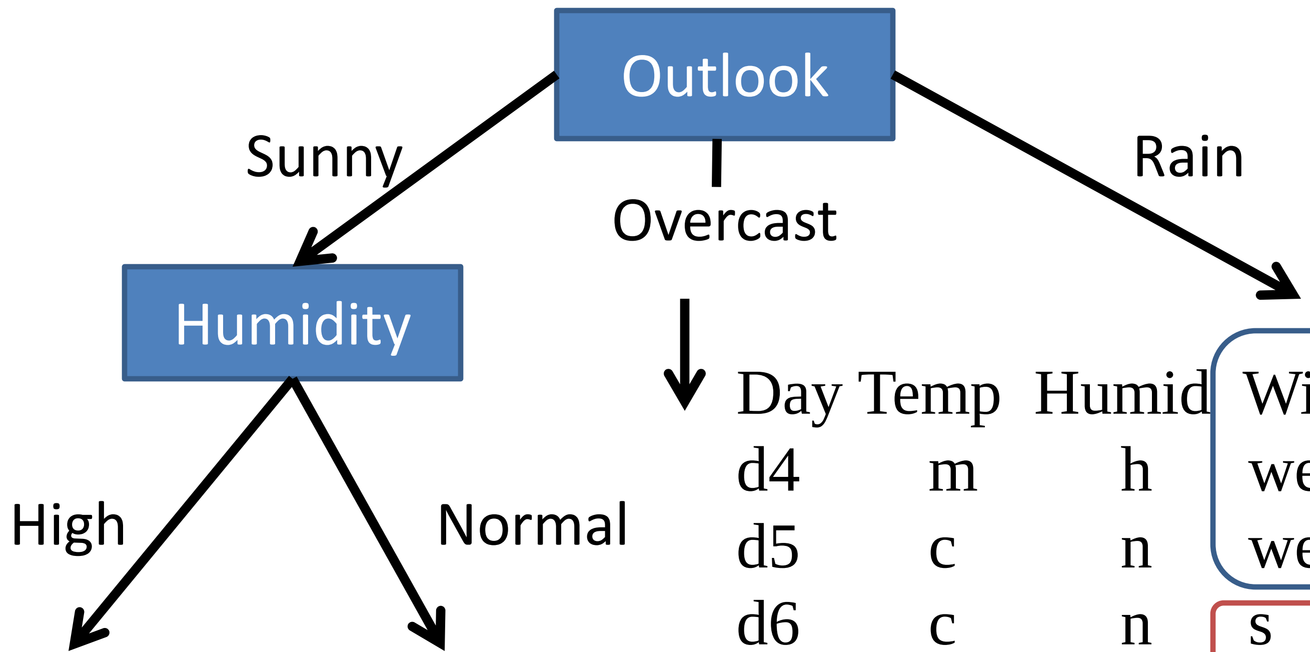
One Step Later

Good day for tennis?



Recurse Again

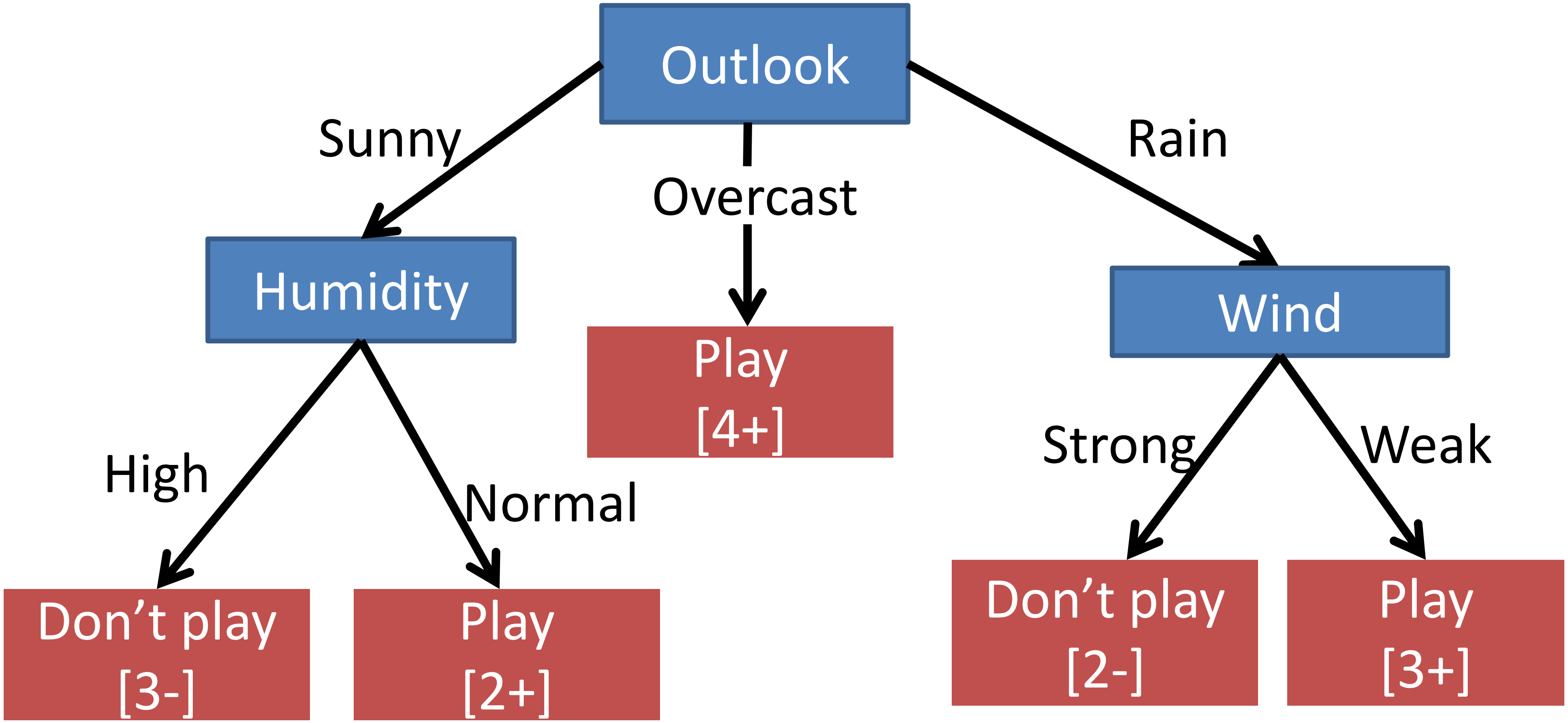
Good day for tennis?



Day	Temp	Humid	Wind	Tennis?
d4	m	h	weak	yes
d5	c	n	weak	yes
d6	c	n	s	n
d10	m	n	weak	yes
d14	m	h	s	n

One Step Later: Final Tree

Good day for tennis?



Issues

- Missing data
- Real-valued attributes
- Many-valued features
- Evaluation
- Overfitting

Missing Data 1

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	$\boxed{?}$	weak	yes
d11	m	n	s	yes

Assign most common value at this node
?=>h

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	$\boxed{?}$	weak	yes
d11	m	n	s	yes

Assign most common value for class
?=>n

Missing Data 2

Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	?	weak	yes
d11	m	n	s	yes

[0.75+, 3-]

[1.25+, 0-]

- 75% h and 25% n
- Use in gain calculations
- Further subdivide if other missing attributes
- Same approach to classify test ex with missing attr
  - Classification is most probable classification
  - Summing over leaves where it got divided

Real-Valued Features

- Discretize?

Wind	25	12	12	11	10	10	8	7	7	7	7	6	6	5
Play	n	y	y	n	y	n	n	y	y	y	y	y	y	n

- Threshold split using observed values?

Wind	8	25	7	6	6	10	12	5	7	7	12	10	7	11
Play	n	n	y	y	y	n	y	n	y	y	y	y	y	n

Wind	25	12	12	11	10	10	8	7	7	7	7	6	6	5
Play	n	y	y	n	y	n	n	y	y	y	y	y	y	n

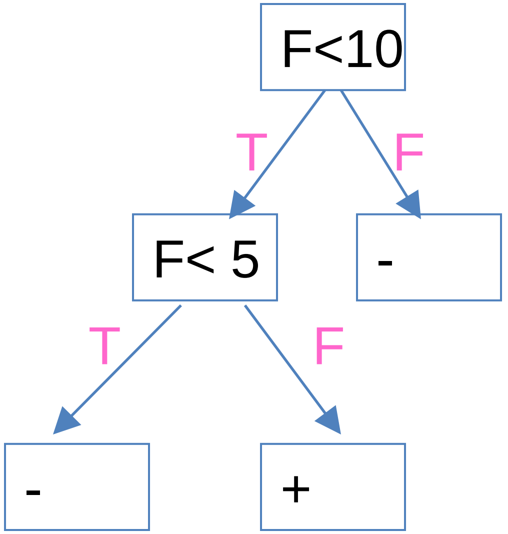
>= 12
Gain = 0.0004

>= 10
Gain = 0.048

Real-Valued Features

Note

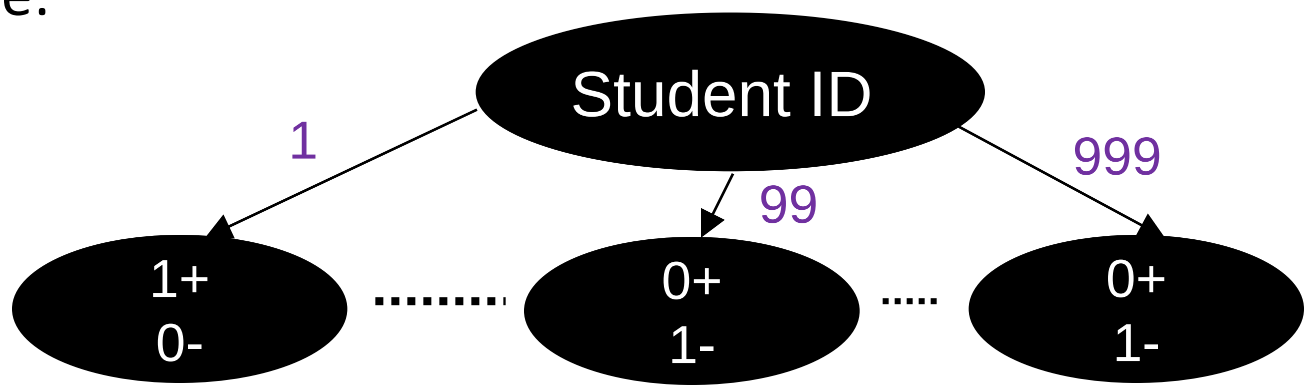
Cannot discard numeric feature after use in one portion of d-tree



Many-Valued Attributes

FAVORS FEATURES WITH HIGH BRANCHING FACTORS
(i.e,. many possible values)

Extreme Case:



At most one example per leaf and all I(.,.) scores for leafs equals zero, so gets perfect score! But generalizes very poorly (i.e., memorizes data)

Fix: Method 1

Convert all features to binary

e.g., Color = {Red, Blue, Green}

From 1 *N*-valued feature to *N* binary features

Color = Red? ⇨ {True, False}

Color = Blue? ⇨ {True, False}

Color = Green? ⇨ {True, False}

Used in Neural Nets and SVMs

D-tree readability probably less, but not necessarily

Fix 2: Gain Ratio

$$\text{Gain Ratio}(S,A) = \quad \text{Gain}(S,A)/\text{SplitInfo}(S,A)$$

$$\text{SplitInfo} = \sum_{v \in \text{Values}(A)} (|S_v| \,/\, |S|) \, \text{Log}_2(|S_v|/|S|)$$

SplitInfo ≅ entropy of S *wrt* values of A

(Contrast with entropy of S *wrt* **target** value)

⇓ attribs with many uniformly distrib values

e.g. if A splits S uniformly into n sets

SplitInformation = $\log_2(n)$… = 1 for Boolean

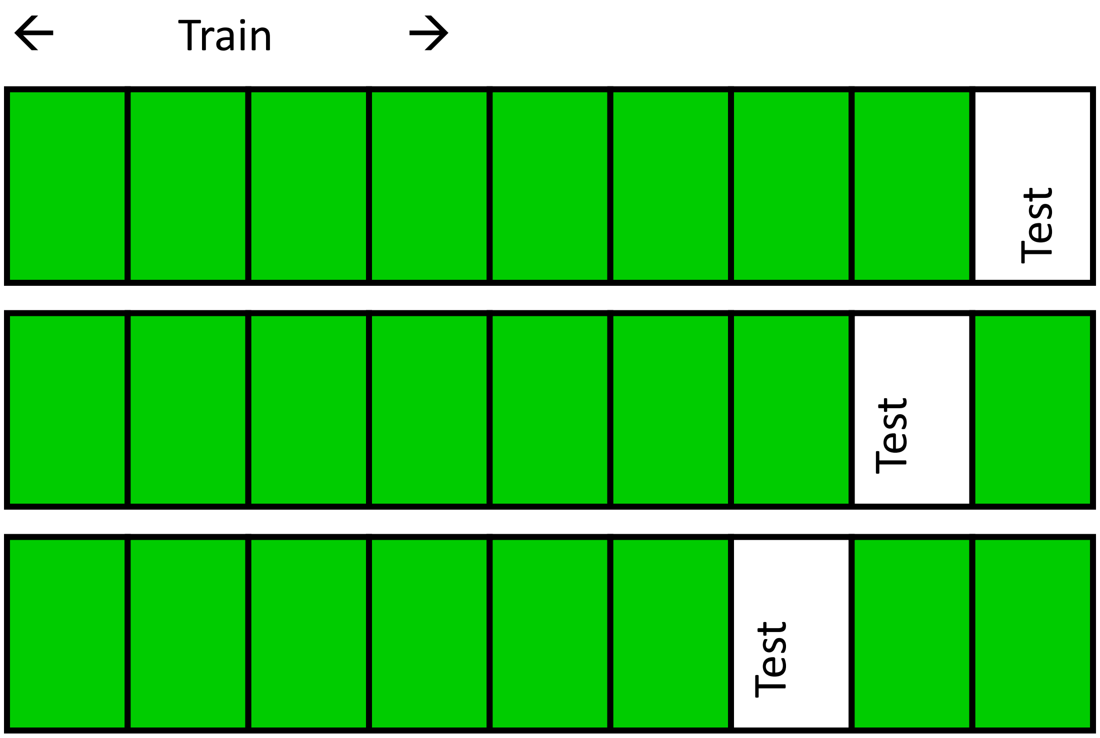
Evaluation

- Question: How well will an algorithm perform on unseen data?

- Cannot score based on training data
  - Estimate will be overly optimistic about algorithm’s performance

Evaluation: Cross Validation

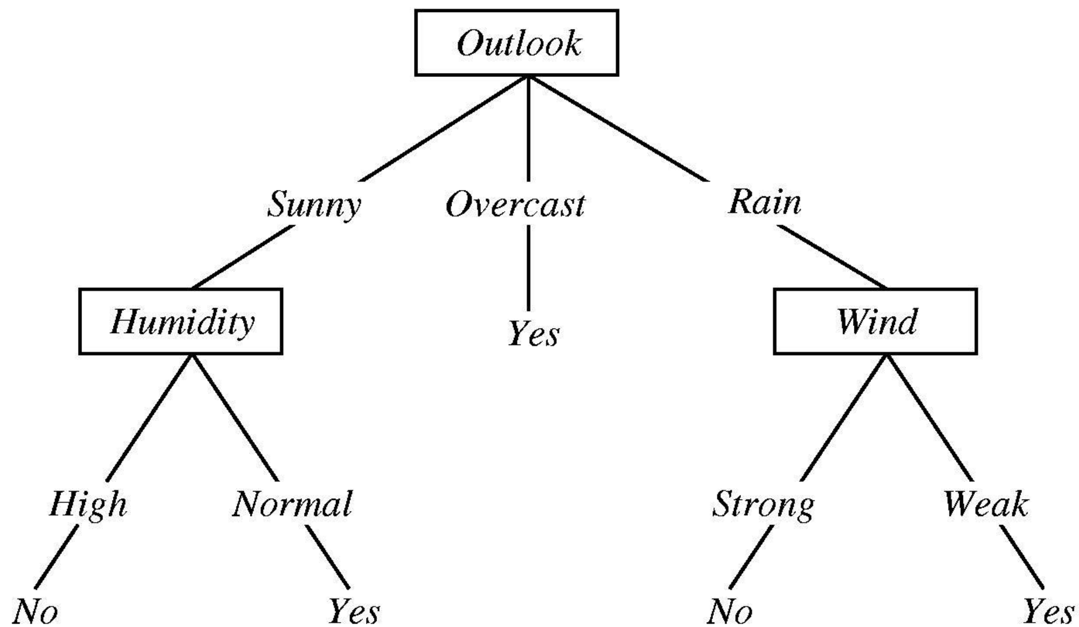
- Partition examples into *k* disjoint sets
- Now create *k* training sets
  - Each set is union of all equiv classes ***except one***
  - So each set has (k-1)/k of the original training data



Cross-Validation (2)

- Leave-one-out
  - Use if < 100 examples (rough estimate)
  - Hold out one example, train on remaining examples
- M of N fold
  - Repeat M times
  - Divide data into N folds, do N fold cross-validation

Overfitting in Decision Trees

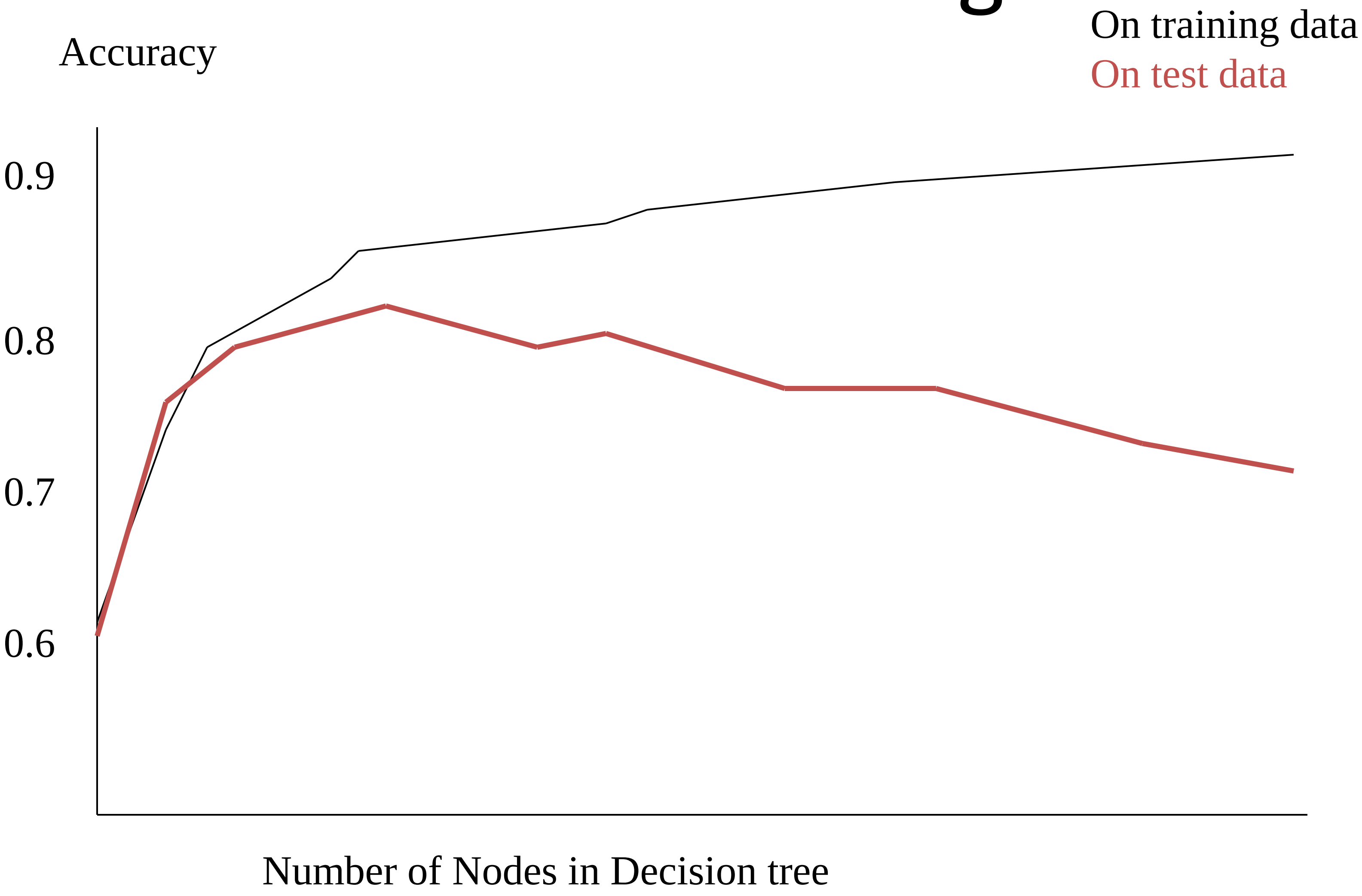


Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Overfitting

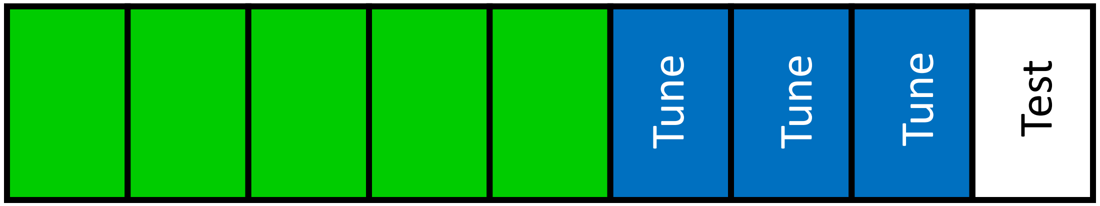


Overfitting Definition

- DT is *overfit* when exists another DT’ and
  - DT has ***smaller*** error on training examples, but
  - DT has ***bigger*** error on test examples
- Causes of overfitting
  - Noisy data, or
  - Training set is too small
- Solutions
  - Reduced error pruning
  - Early stopping
  - Rule post pruning

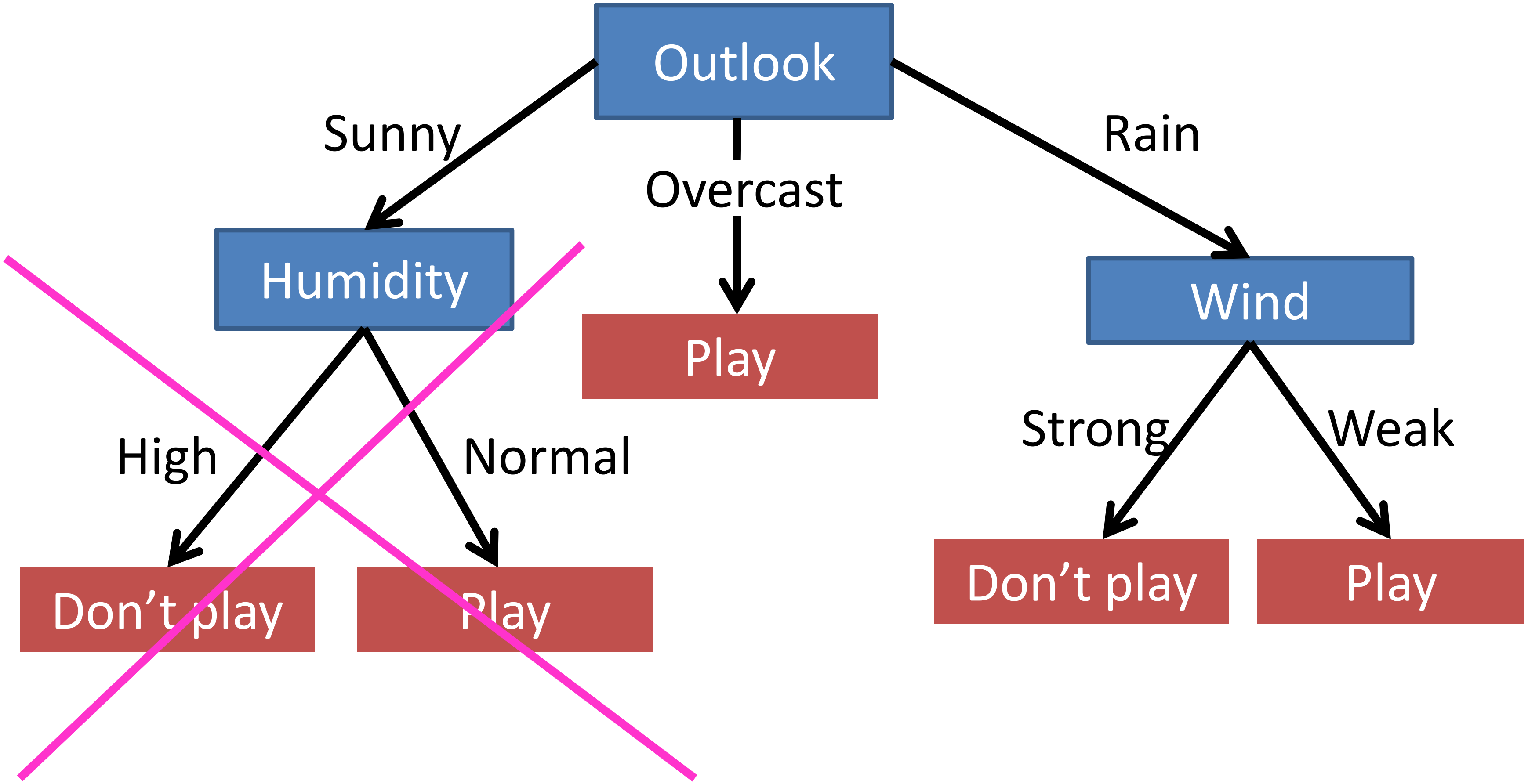
Reduced Error Pruning

- Split data into train and validation set



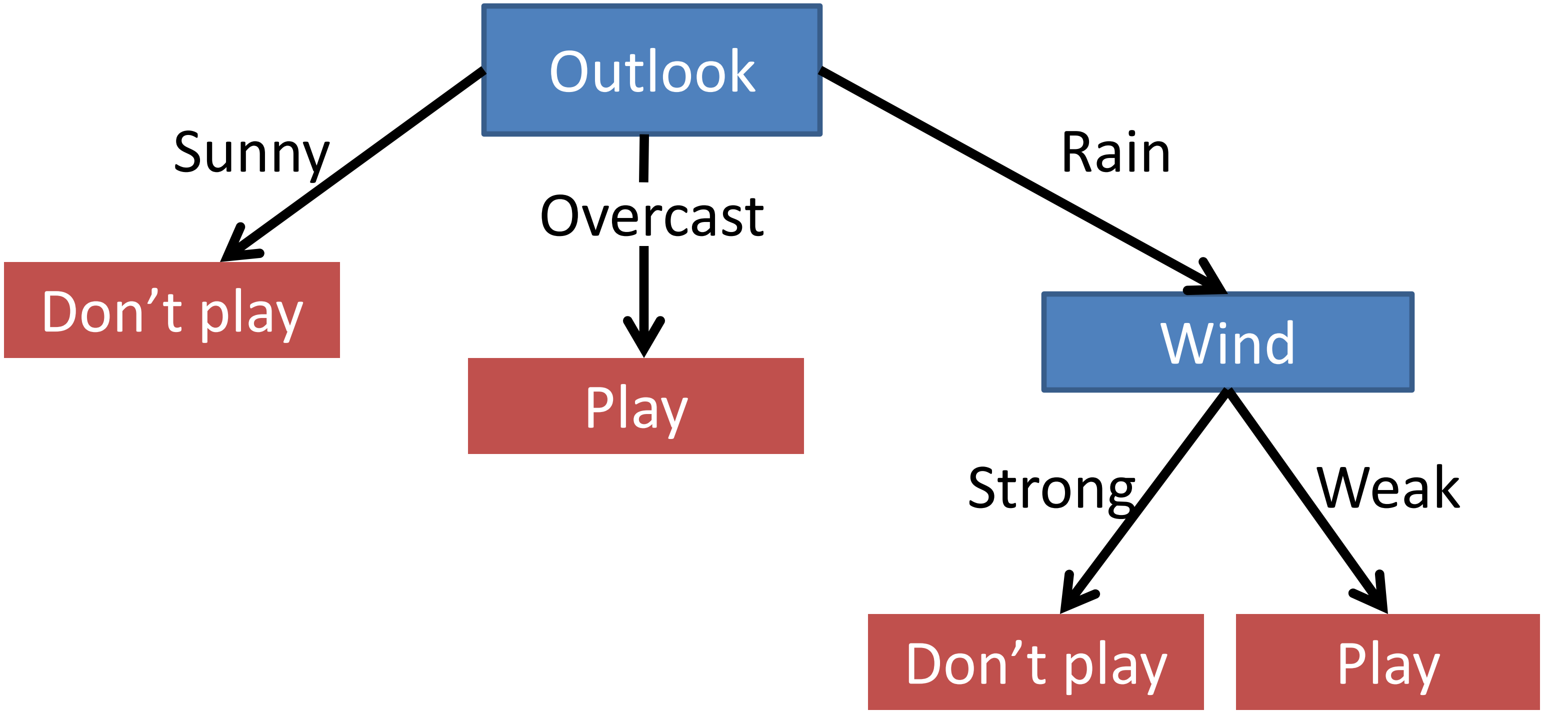
- Repeat until pruning is harmful
  - Remove each subtree and replace it with majority class and evaluate on validation set
  - Remove subtree that leads to largest gain in accuracy

Reduced Error Pruning Example



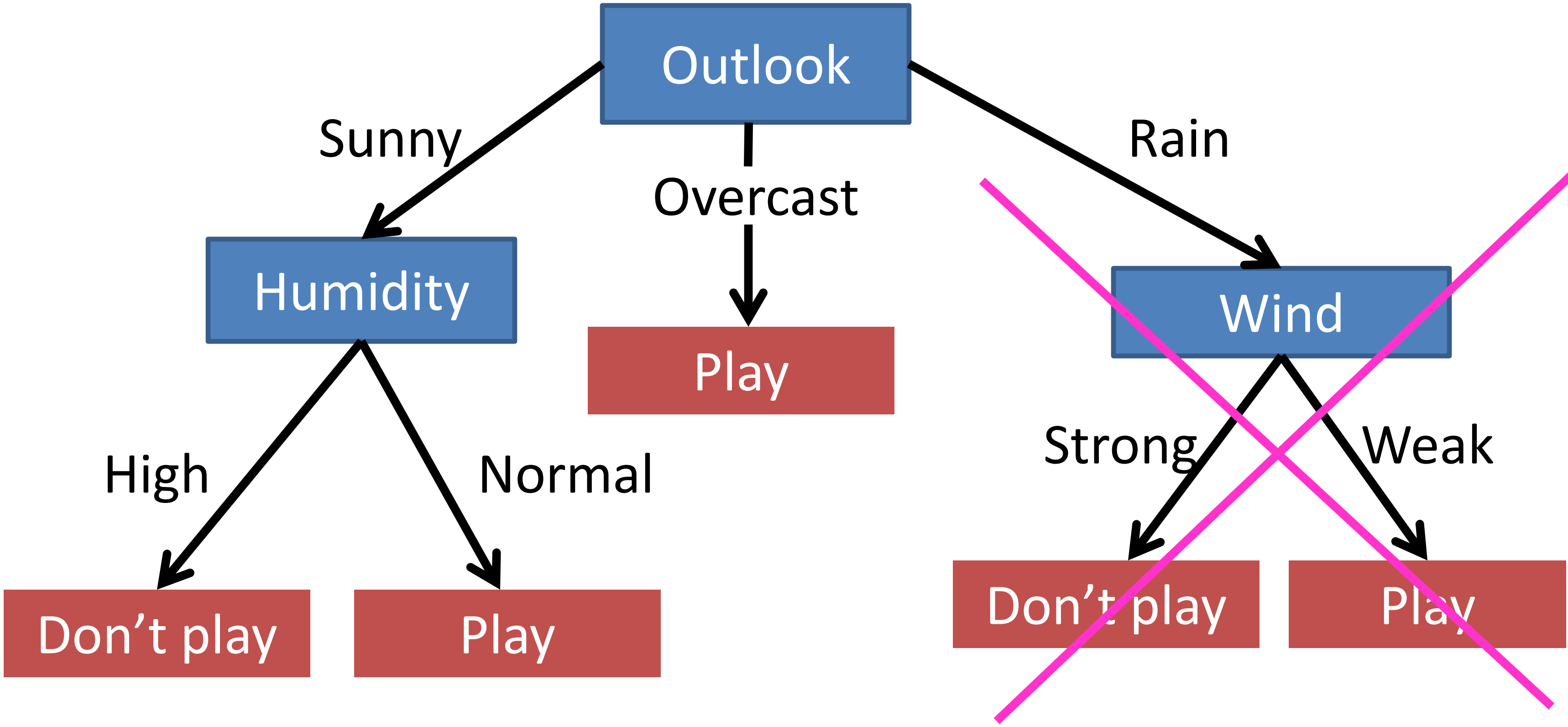
Validation set accuracy = 0.75

Reduced Error Pruning Example

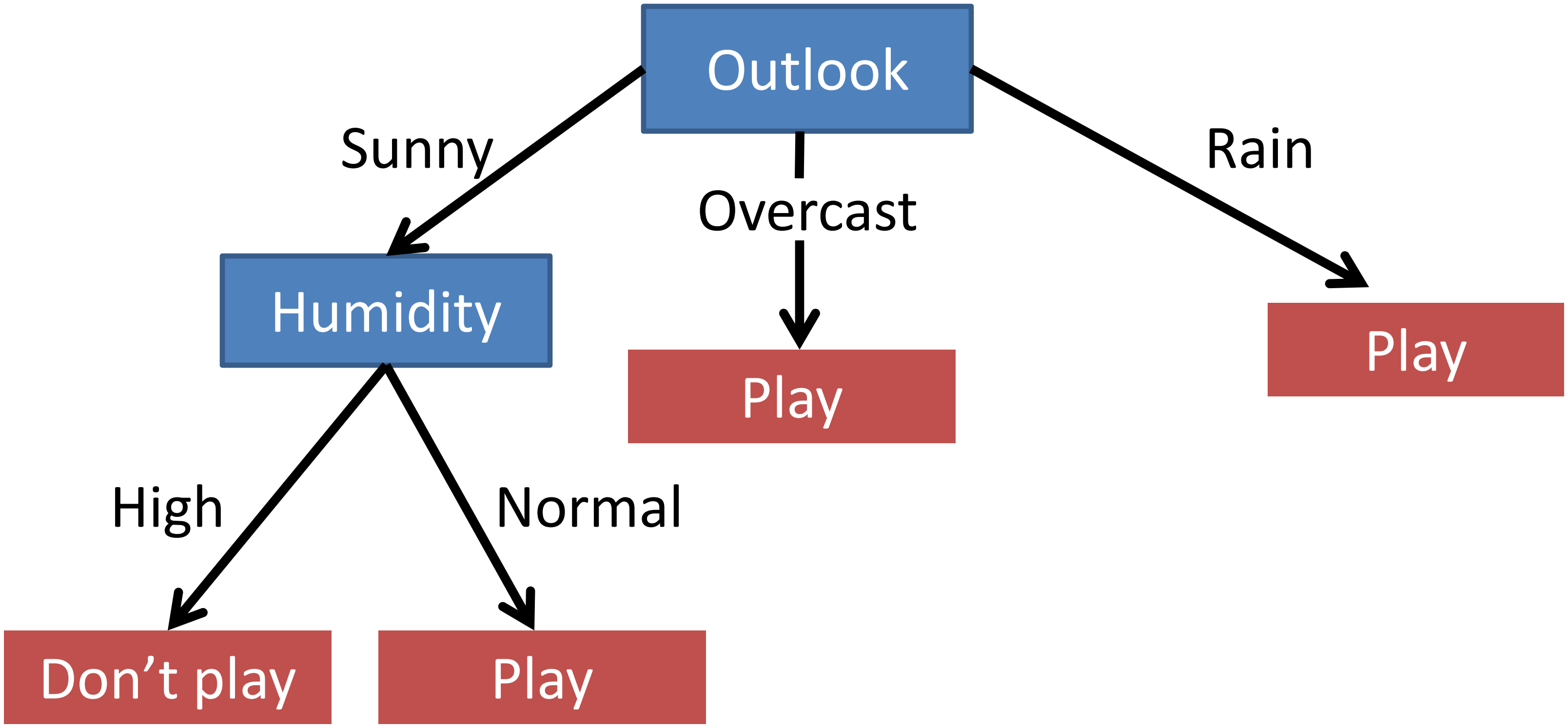


Validation set accuracy = 0.80

Reduced Error Pruning Example

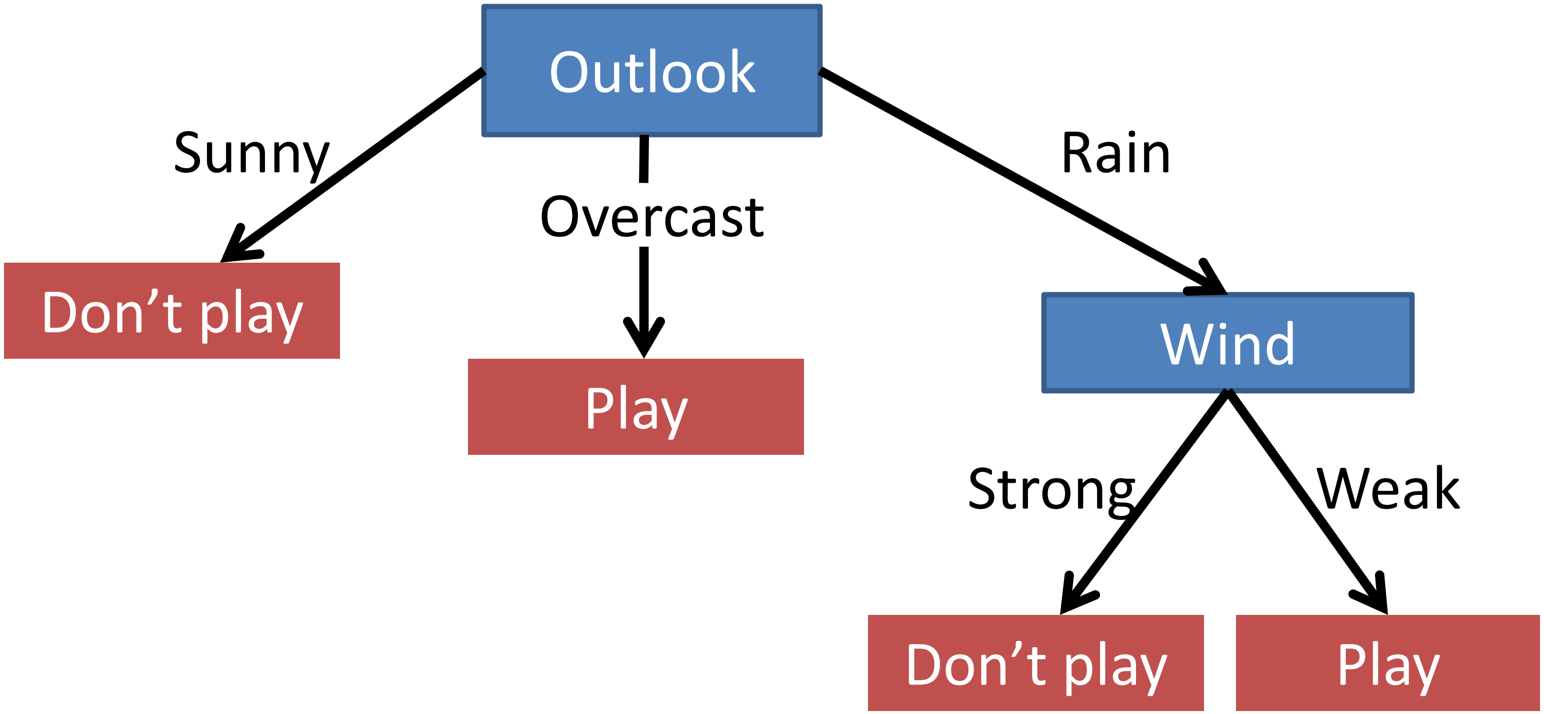


Reduced Error Pruning Example



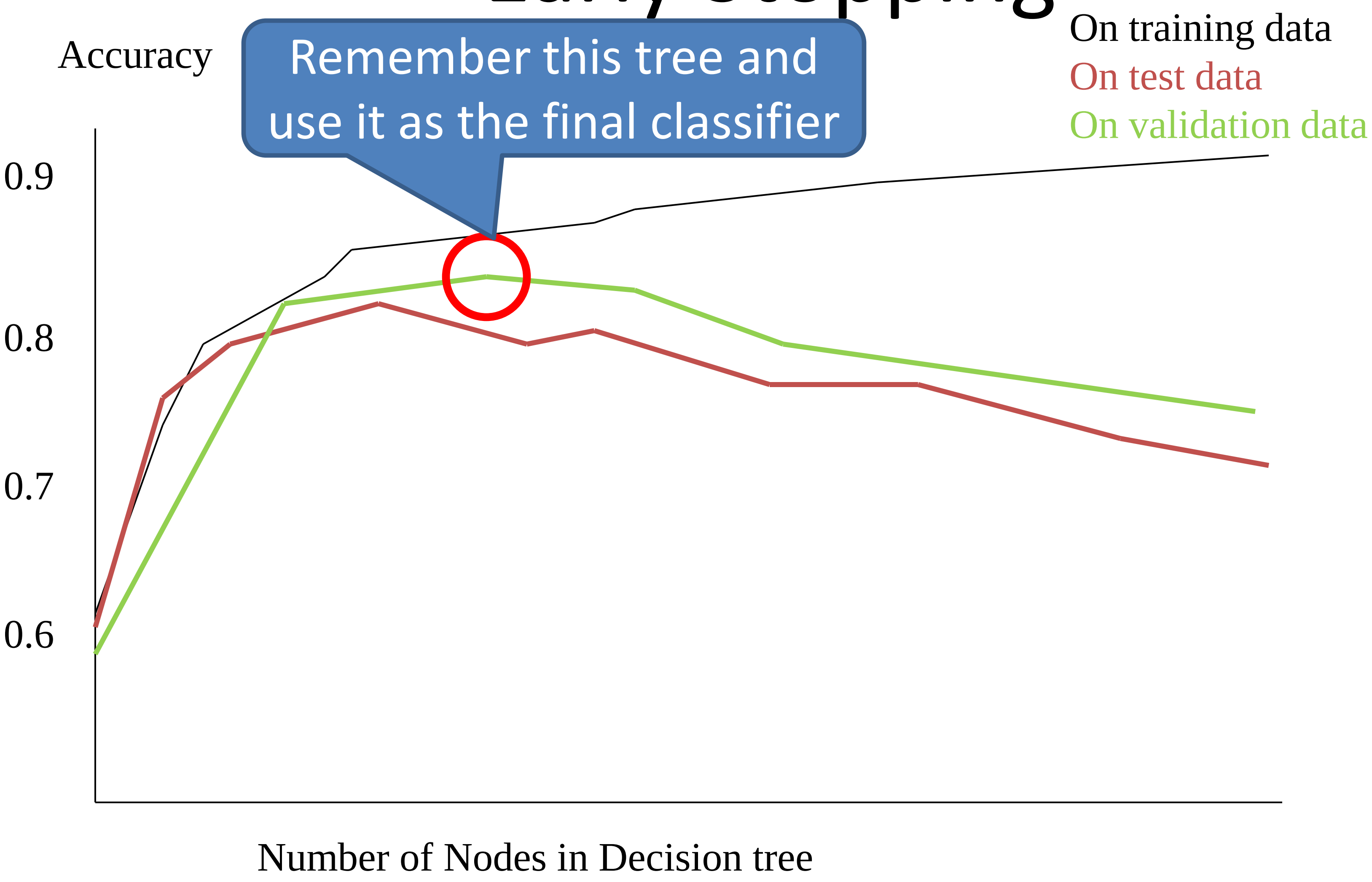
Validation set accuracy = 0.70

Reduced Error Pruning Example



Use this as final tree

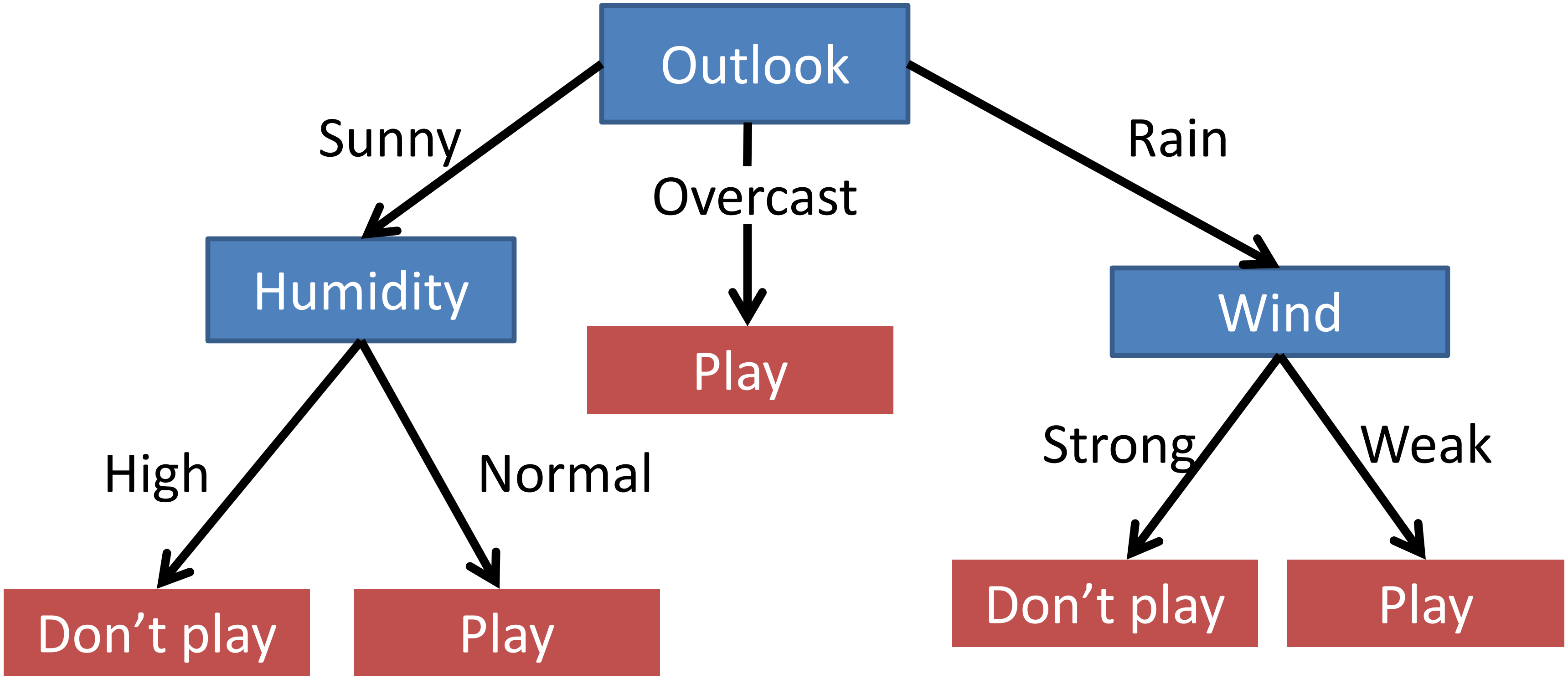
Early Stopping



Post Rule Pruning

- Split data into train and validation set
- Prune each rule independently
  - Remove each pre-condition and evaluate accuracy
  - Pick pre-condition that leads to largest improvement in accuracy
- Note: ways to do this using training data and statistical tests

Conversion to Rule



$\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{High} \Rightarrow \text{Don't play}$

$\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal} \Rightarrow \text{Play}$

$\text{Outlook} = \text{Overcast} \Rightarrow \text{Play}$

$\ldots$

Example

$\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{High} \Rightarrow \text{Don't play}$

Validation set accuracy = 0.68

$\rightarrow \text{Outlook} = \text{Sunny} \Rightarrow \text{Don't play}$ Validation set accuracy = 0.65

$\rightarrow \boxed{\text{Humidity} = \text{High} \Rightarrow \text{Don't play}}$ Validation set accuracy = 0.75

Keep this rule

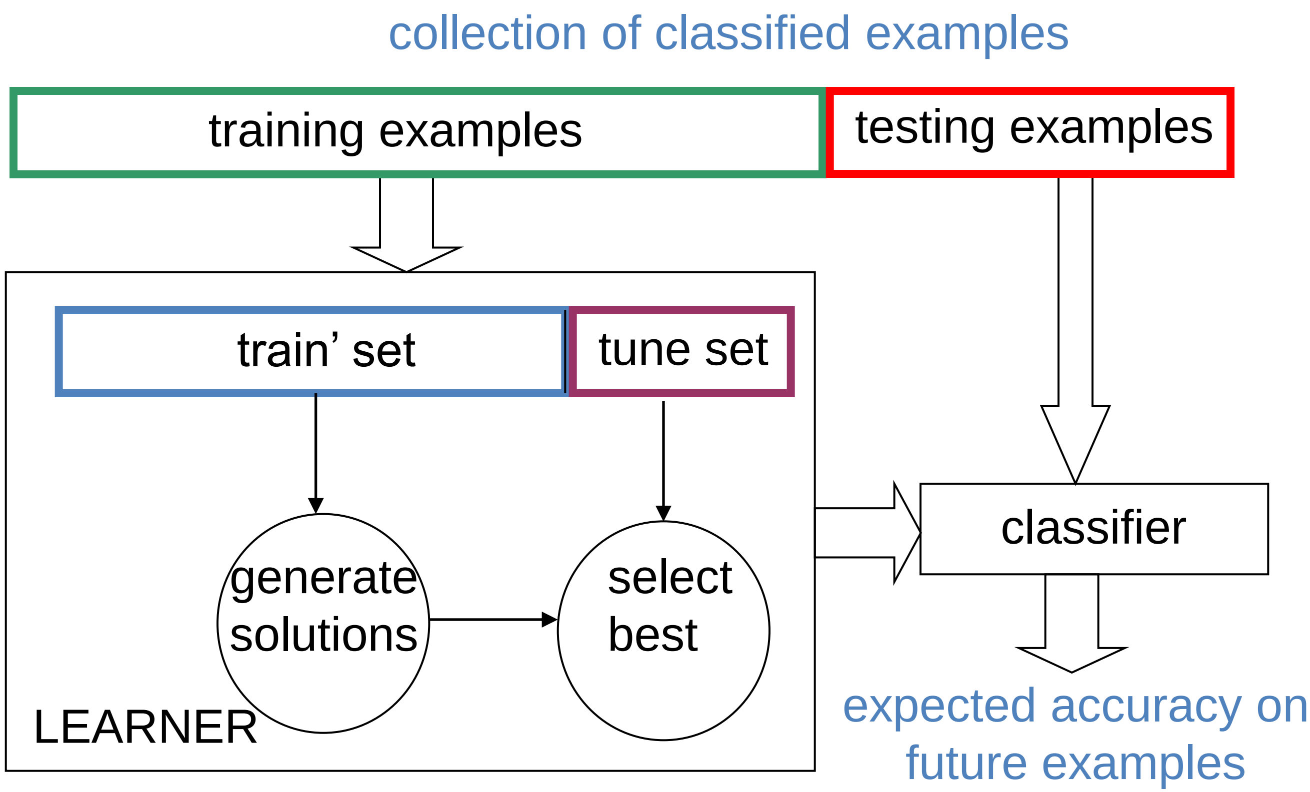
15 Minute Break

Outline

- Decision Trees

- Experimental Methodology
  - Methodology overview
  - How to present results
  - Hypothesis testing

Experimental Methodology: A Pictorial Overview



Statistical techniques such as 10-fold cross validation and *t*-tests are used to get meaningful results

Using <u>Tuning</u> Sets

- Often, an ML system has to choose when to stop learning, select among alternative answers, etc.
- One wants the model that produces the highest accuracy on **future** examples ("overfitting avoidance")
- It is a **"cheat"** to look at the **test** set while still learning
- Better method
  - Set aside part of the training set
  - Measure performance on this "tuning" data to estimate future performance for a given set of parameters
  - Use best parameter settings, train with **all** training data (except **test** set) to estimate future performance on **new** examples

Proper Experimental Methodology Can Have a Huge Impact!

A 2002 paper in *Nature* (a major, major journal) needed to be corrected due to “training on the testing set”

Original report : 95% accuracy (5% error rate)

Corrected report (which still is buggy):
73% accuracy (27% error rate)

Error rate increased over 400%!!!

Parameter Setting

Notice that each train/test fold may get <u>different</u> parameter settings!

- That's fine (and proper)

I.e. , a "parameterless"* algorithm internally sets parameters for **each data set** it gets

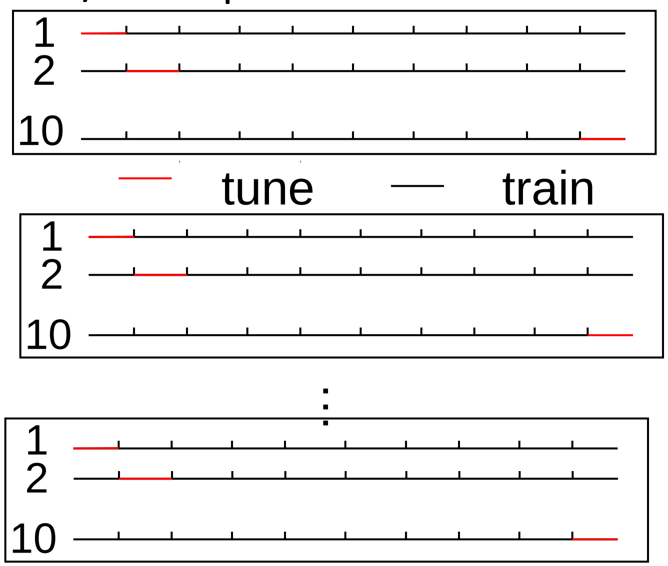
Using Multiple Tuning Sets

- Using a **single** tuning set can be unreliable predictor, plus some data "wasted"
  Hence, often the following is done:
  1) For each possible set of parameters,
     a) Divide <u>training</u> data into **train'** and **tune** sets, using **N-fold cross validation**
     b) Score this set of parameter value, average **tune** set accuracy
  2) Use **best** set of parameter settings and **<u>all</u> (train' + tune)** examples
  3) Apply resulting model to **test** set

Tuning a Parameter
- Sample Usage

Step1: Try various values for *k* (e.g., neighborhood size/distance function in k-NN

Use 10 train/tune splits for each *k*



K=0 — Tune set accuracy (ave. over 10 runs)=92%

K=2 — Tune set accuracy (ave. over 10 runs)=97%

K=100 — Tune set accuracy (ave. over 10 runs)=80%

Step2: Pick best value for *k* (eg. *k* = 2),
Then train using **<u>all</u>** <u>training data</u>

Step3: Measure accuracy on **test set**

What to Do for the FIELDED System?

- Do **not** use any **test** sets
- Instead only use **tuning** sets to determine good parameters
  - **Test** sets used to estimate **future** performance
  - You can report this estimate to your "customer," then use **all** the data to retrain a "product" to give them

What's Wrong with This?

1. Do a cross-validation study to set parameters
2. Do another cross-validation study, using the best parameters, to estimate future accuracy
   - How will this relate to the "true" future accuracy?
   - Likely to be an overestimate

What about

1. Do a proper train/tune/test experiment
2. Improve your algorithm; goto 1

   (Machine Learning's "dirty little" secret!)

Why Not Learn After Each Test Example?

- In "production mode," this would make sense (assuming one received the correct label)
- In "experiments," we wish to estimate

  Probability we'll label the next example correctly

  need several samples to accurately estimate

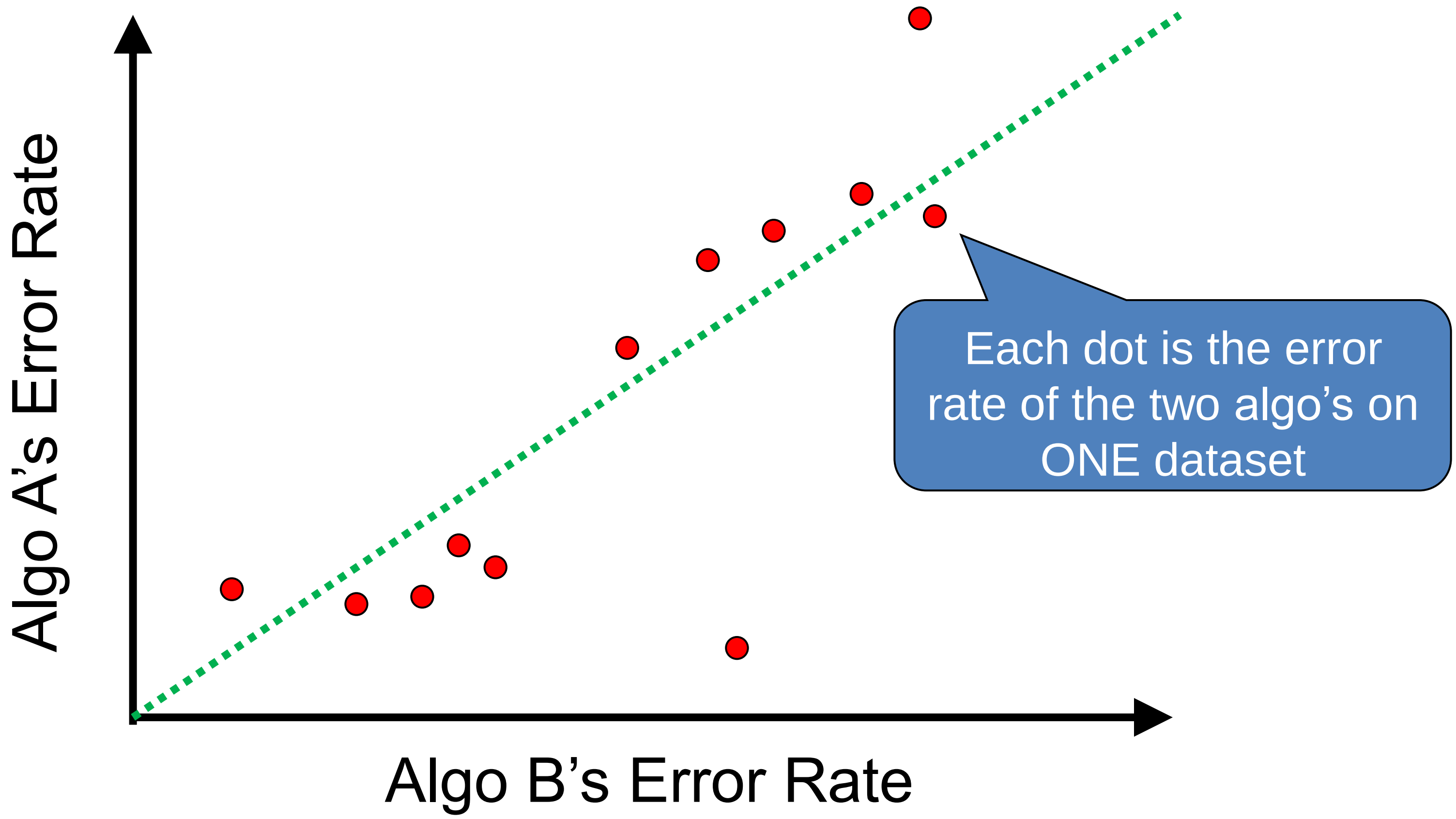
Outline

- Decision Trees

- Experimental Methodology
  - Methodology overview
  - How to present results
  - Hypothesis testing

Scatter Plots

- Compare Two Algo's on Many Datasets



Evaluation Metrics

Called a confusion matrix or contingency table

	Predicted True	Predicted False
Actually True	TP	FN
Actually False	$\boxed{\text{FP}}$	TN

The number of times true is "confused" with false by the algorithm

ROC Curves

- ROC: Receiver Operating Characteristics
- Started during radar research during WWII
- Judging algorithms on accuracy alone may not be good enough when getting a positive wrong costs more than getting a negative wrong (or vice versa)
  - Eg, medical tests for serious diseases
  - Eg, a movie-recommender (ala' NetFlix) system

Evaluation Metrics

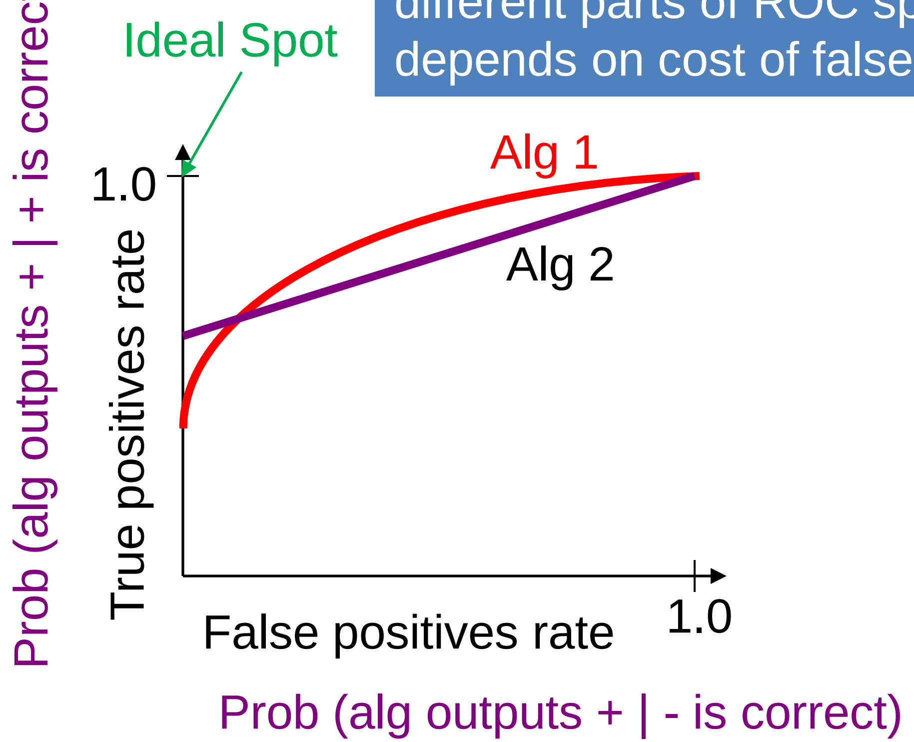
$$\text{True positive rate (tpr)} = \frac{TP}{TP + FN}$$

$$\text{False positive rate (fpr)} = \frac{FP}{TN + FP}$$

	Predicted True	Predicted False
Actually True	TP	FN
Actually False	FP	TN

ROC Curves Graphically

Different algorithms can work better in different parts of ROC space. This depends on cost of false + vs false -



Creating an ROC Curve
- the Standard Approach

- You need an ML algorithm that outputs NUMERIC results such as prob(example is +)
- You can use ensembles (later) to get this from a model that only provides Boolean outputs
  - Eg, have 100 models vote & count votes

Algorithm for Creating ROC Curves

Step 1: Sort predictions on test set

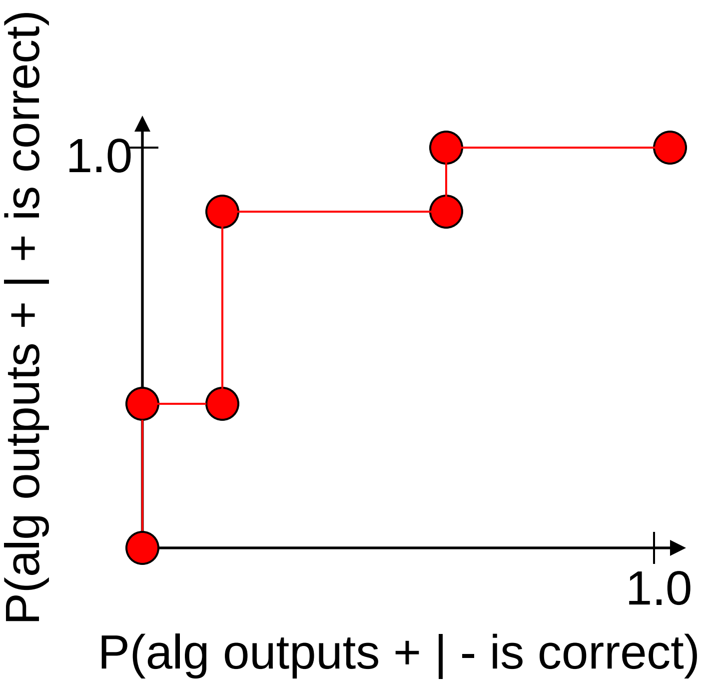
Step 2: Locate a threshold between examples with opposite categories

Step 3: Compute TPR & FPR for each threshold of Step 2

Step 4: Connect the dots

Plotting ROC Curves - Example

ML Algo Output (Sorted)			Correct Category
Ex 9	.99		+
Ex 7	.98	TPR=(2/5), FPR=(0/5)	+
Ex 1	.72	TPR=(2/5), FPR=(1/5)	-
Ex 2	.70		+
Ex 6	.65	TPR=(4/5), FPR=(1/5)	+
Ex 10	.51		-
Ex 3	.39	TPR=(4/5), FPR=(3/5)	-
Ex 5	.24	TPR=(5/5), FPR=(3/5)	+
Ex 4	.11		-
Ex 8	.01	TPR=(5/5), FPR=(5/5)	-

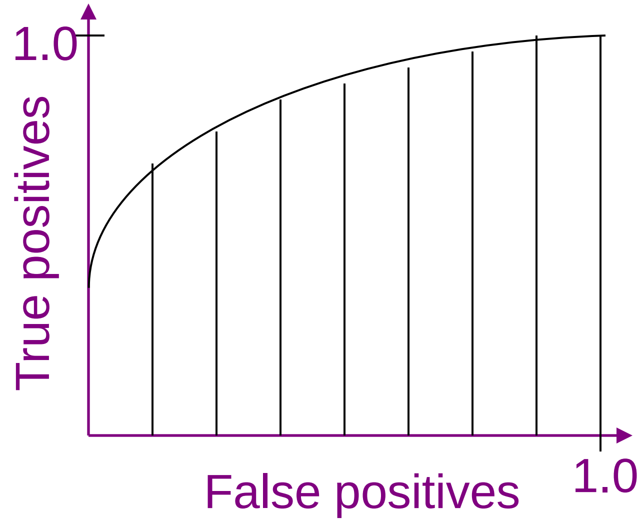


ROC's and Many Models (<u>not</u> in the ensemble sense)

- It is not necessary that we learn <u>*one*</u> model and then threshold its output to produce an ROC curve
- You could learn <u>different models</u> for <u>different regions</u> of ROC space
- For example, see Goadrich, Oliphant, & Shavlik ILP '04 and MLJ '06

Area Under ROC Curve

A common metric for experiments is to <u>numerically integrate</u> the ROC Curve



AUC = Wilcoxon-Mann-Whitney Statistic

ROC's & Skewed Data

- One strength of ROC curves is that they are a good way to deal with skewed data
(|+| >> |-|) since the axes are fractions (rates) independent of the # of examples
- You must be careful though!
- Low FPR * (many negative ex)
  = sizable number of FP
- Possibly more than # of TP

Evaluation Metrics: Precision and Recall

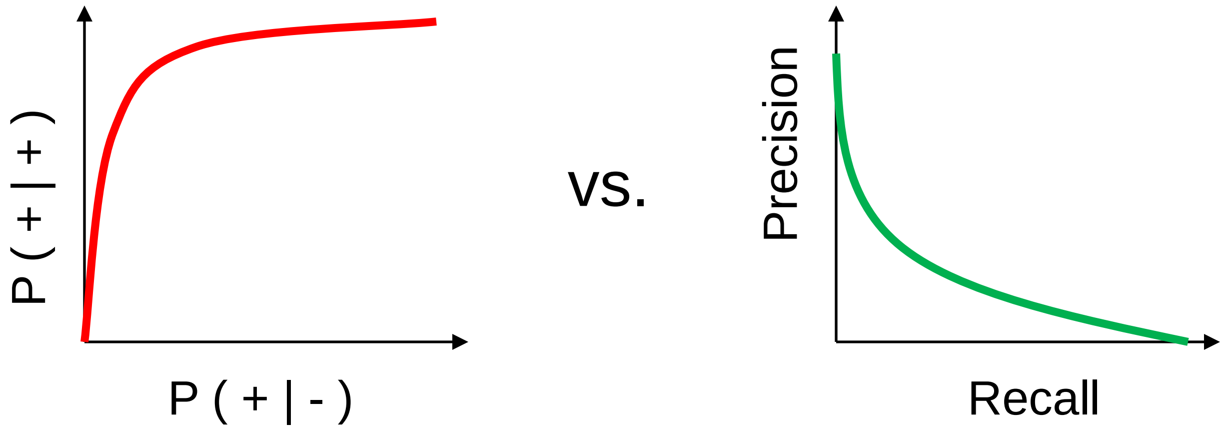
$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

	Predicted True	Predicted False
Actually True	TP	FN
Actually False	FP	TN

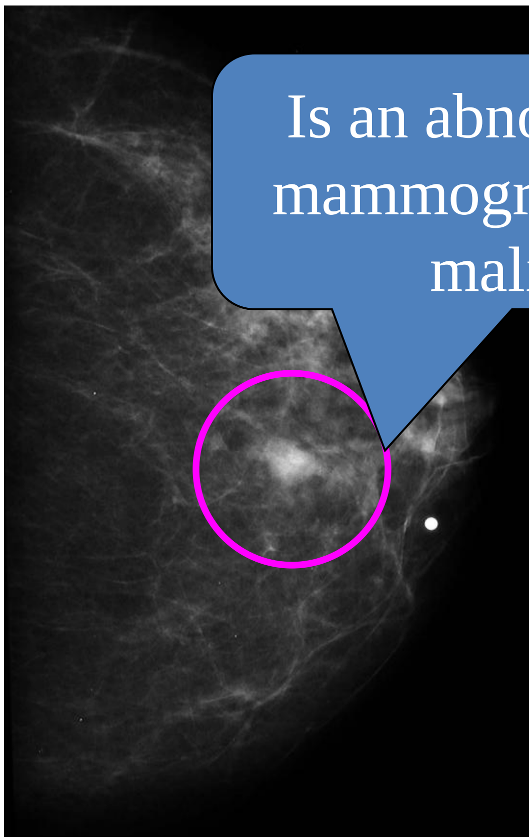
ROC vs. Recall-Precision

You can get very different visual results on the same data



The reason for this is that there may be lots of – ex's
(eg, might need to include 100 neg's to get 1 more pos)

Two Highly Skewed Domains

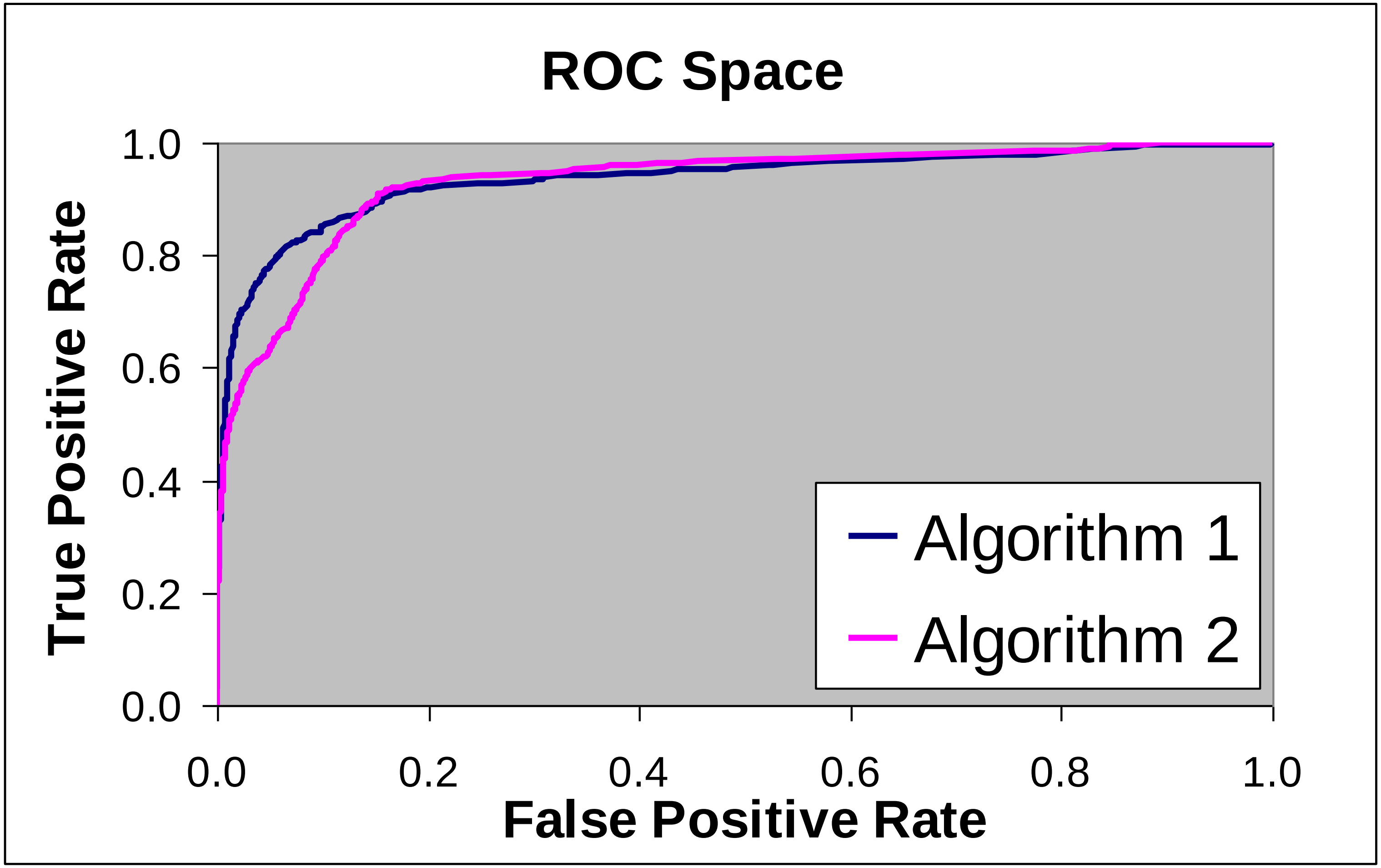


Is an abnormality on a mammogram benign or malignant?

Do these two identities refer to the same person?

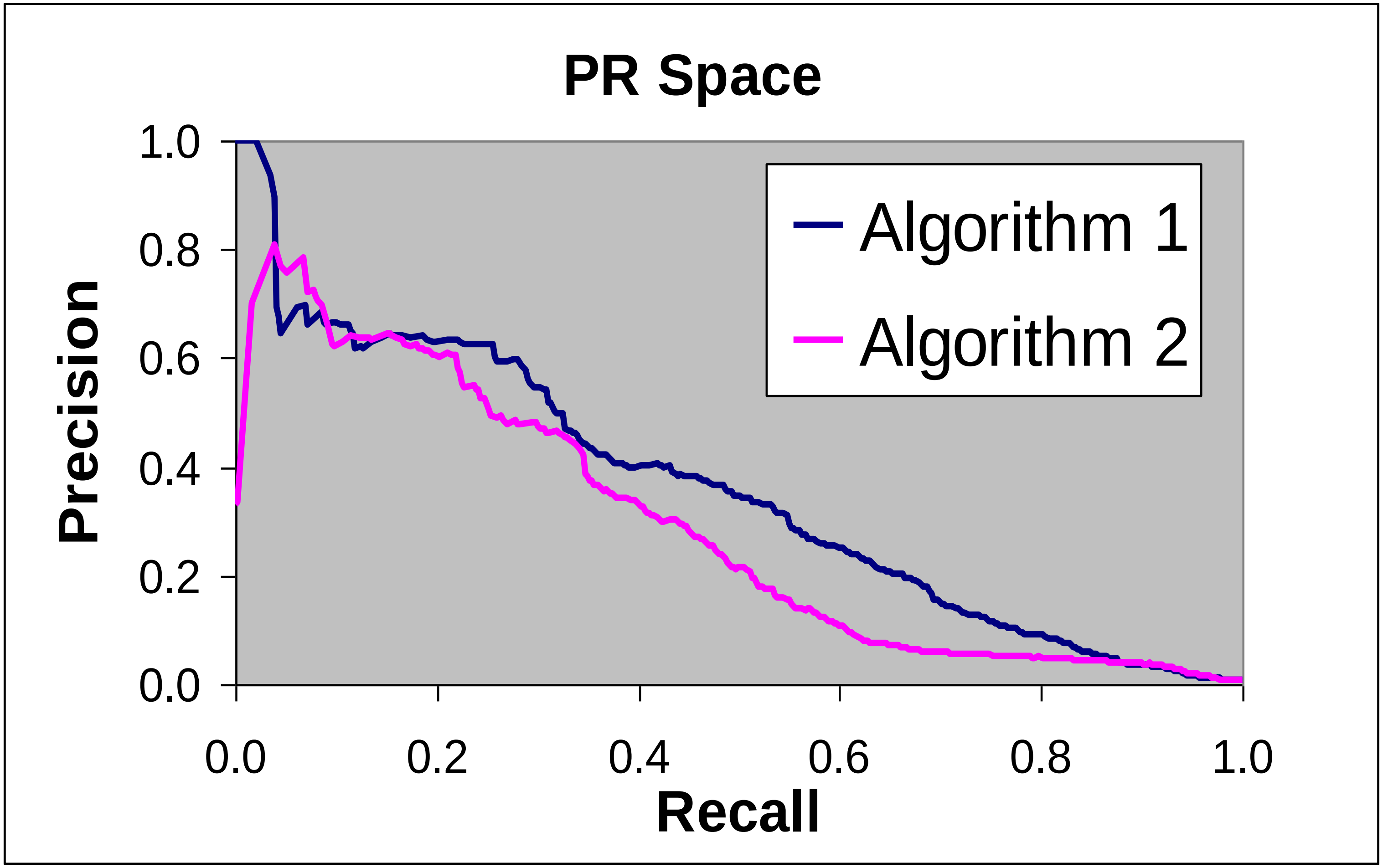
Diagnosing Breast Cancer

[Real Data: Davis et al. IJCAI 2005]



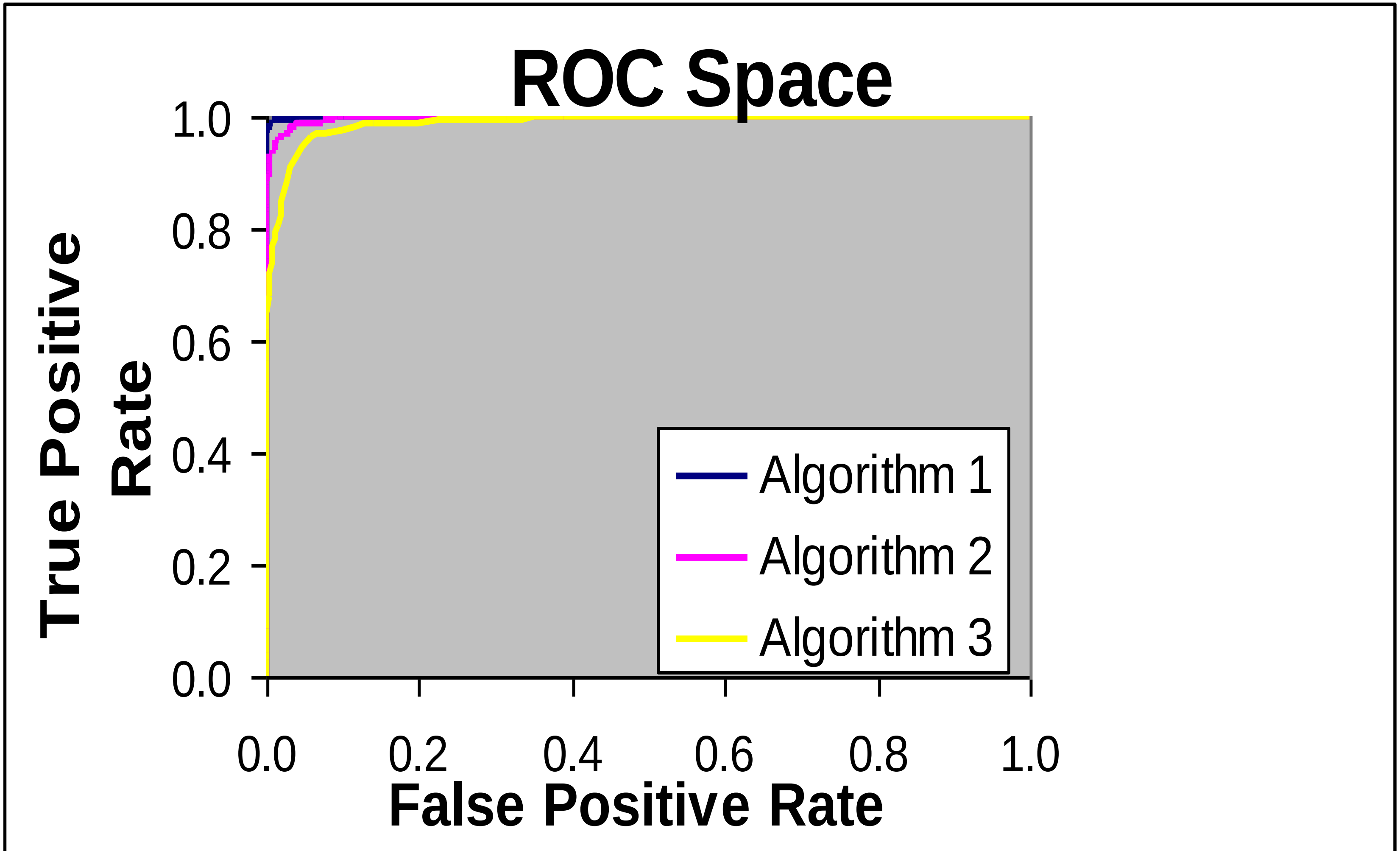
Diagnosing Breast Cancer

[Real Data: Davis et al. IJCAI 2005]



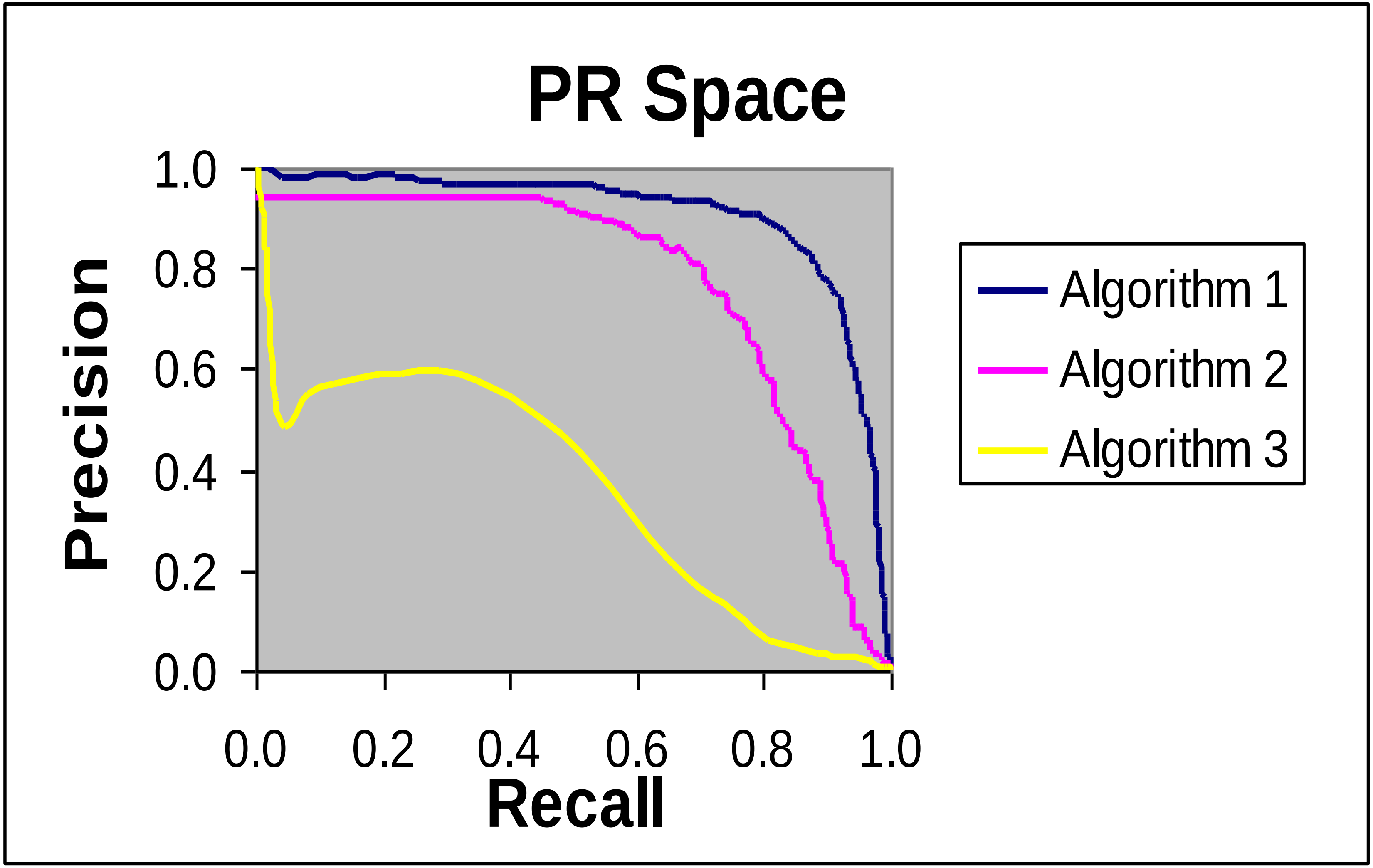
Predicting Aliases

[Synthetic data: Davis et al. ICIA 2005]



Predicting Aliases

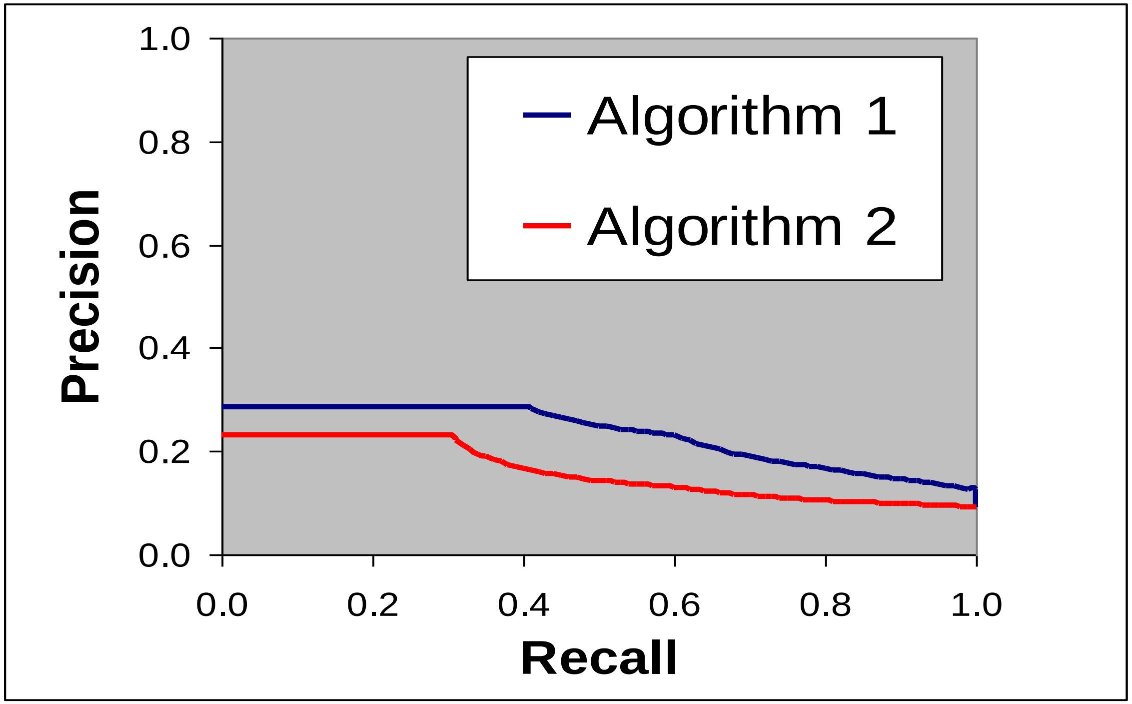
[Synthetic data: Davis et al. ICIA 2005]



Four Questions about PR space and ROC space

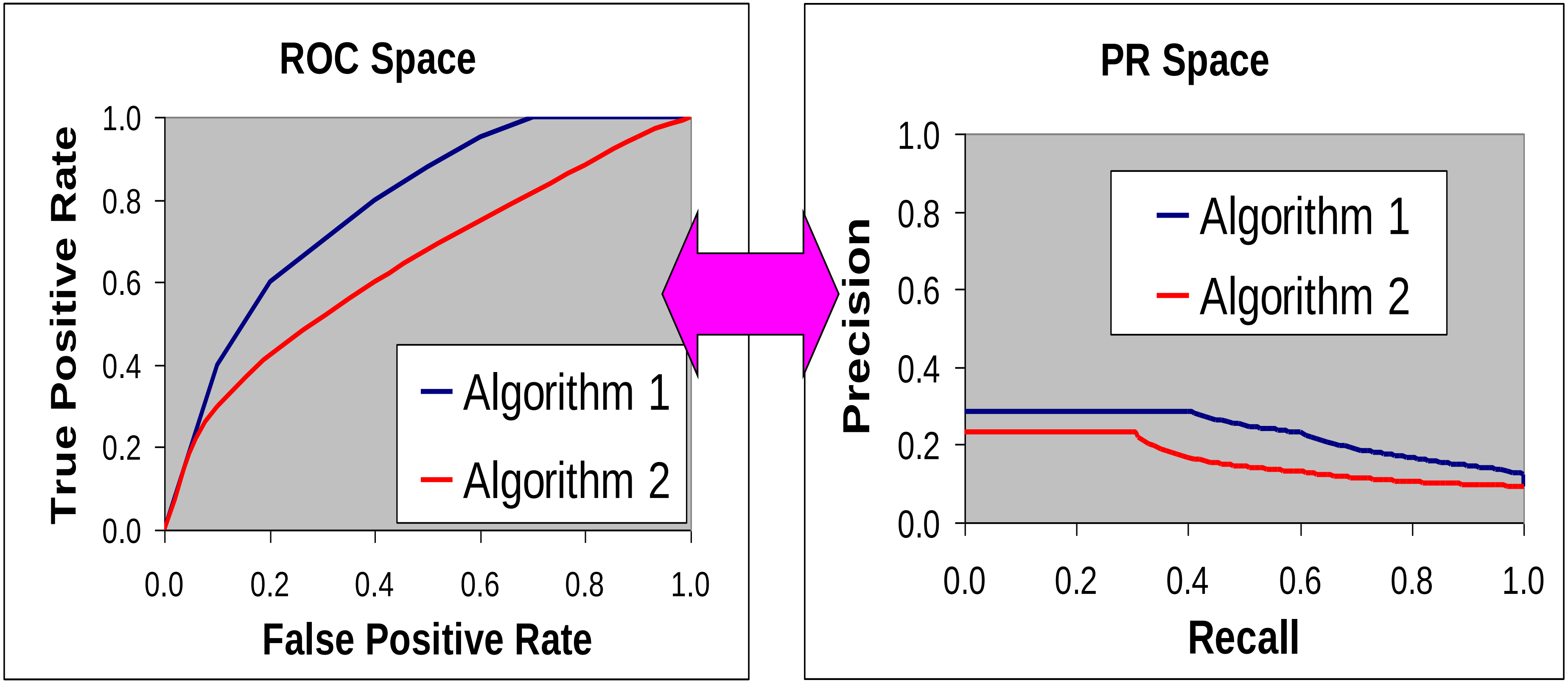
- Q1: If a curve dominates in one space will it dominate in the other?
- Q2: What is the "best" PR curve?
- Q3: How do you interpolate in PR space?
- Q4: Does optimizing AUC in one space optimize it in the other space?

Definition: Dominance



A1: Dominance Theorem

For a fixed number of positive and negative examples, one curve dominates another curve in ROC space if and only if the first curve dominates the second curve in PR space



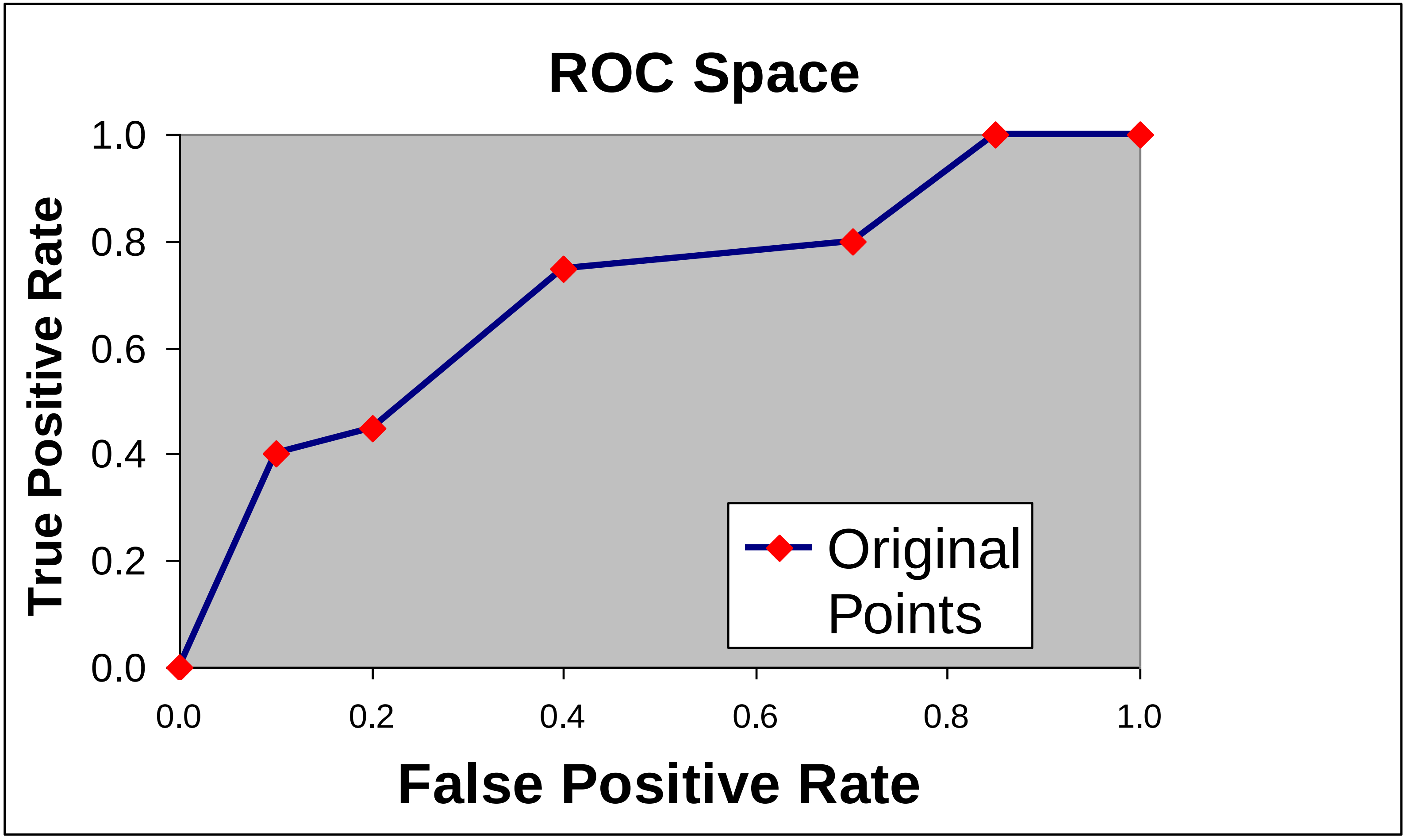
Q2: What is the "best" PR curve?

- The "best" curve in ROC space for a set of points is the convex hull [Provost et al '98]
  - It is achievable
  - It maximizes AUC

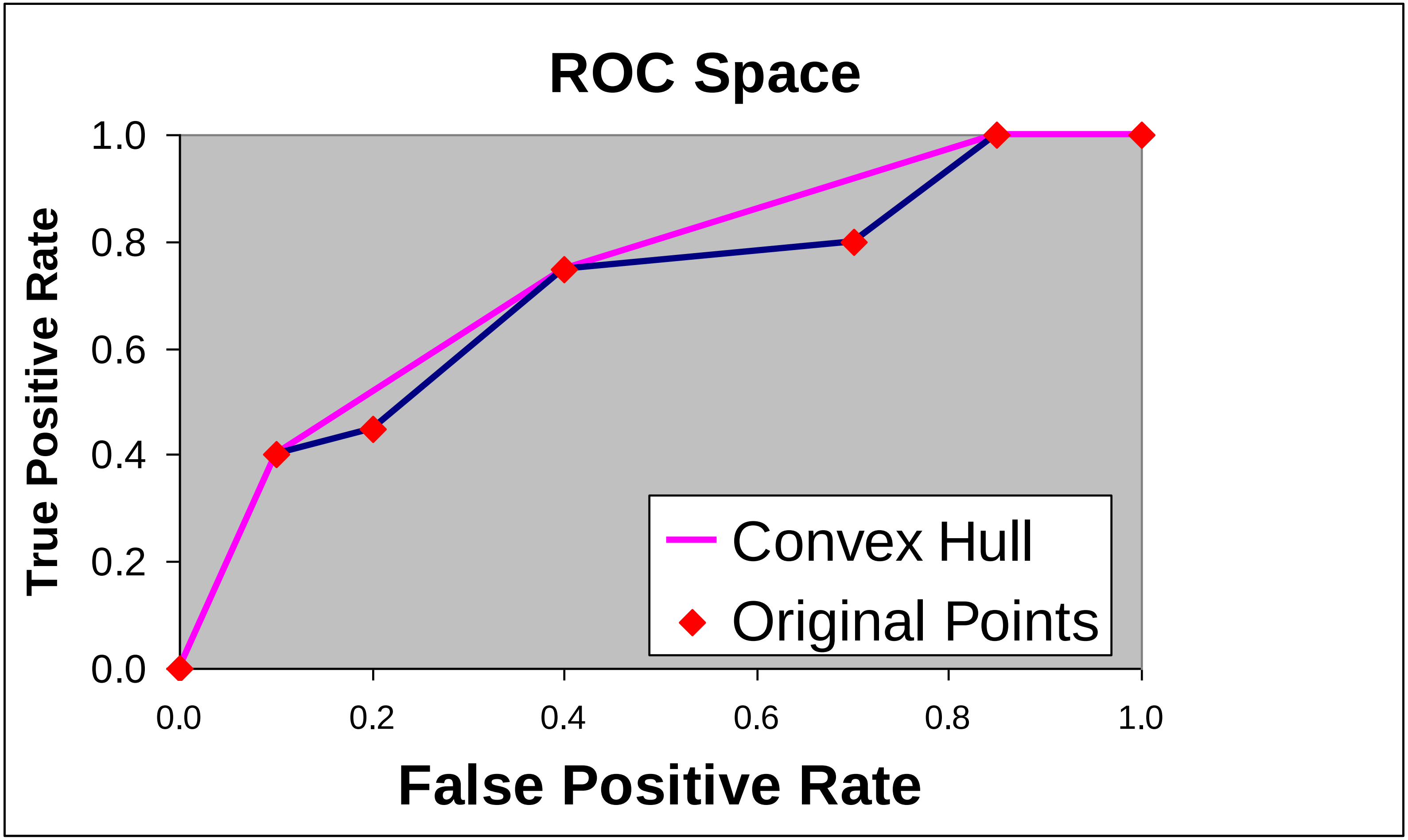
Q: Does an analog to convex hull exist in PR space?

A2: Yes! We call it the **Achievable PR Curve**

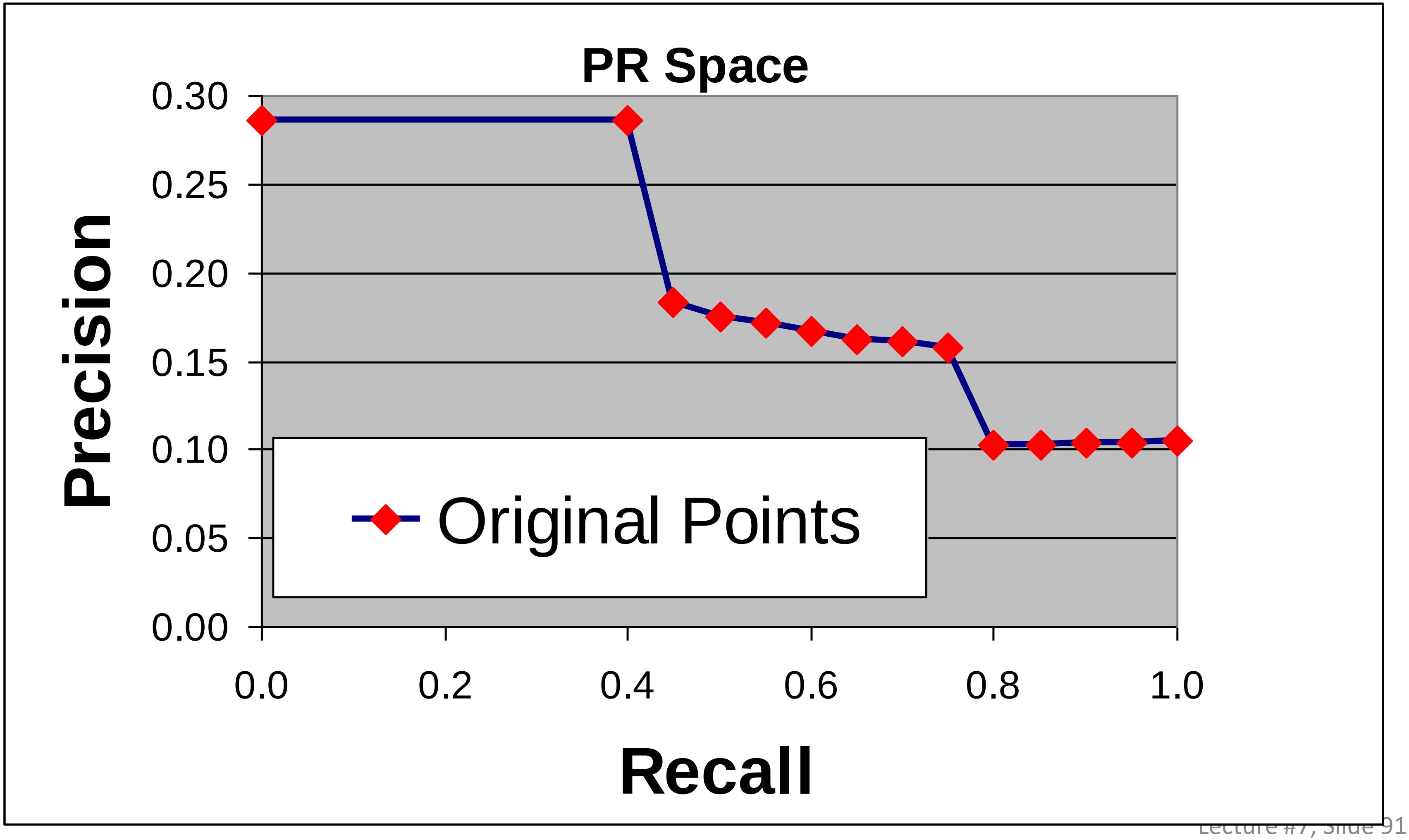
Convex Hull


ROC Space
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Original Points

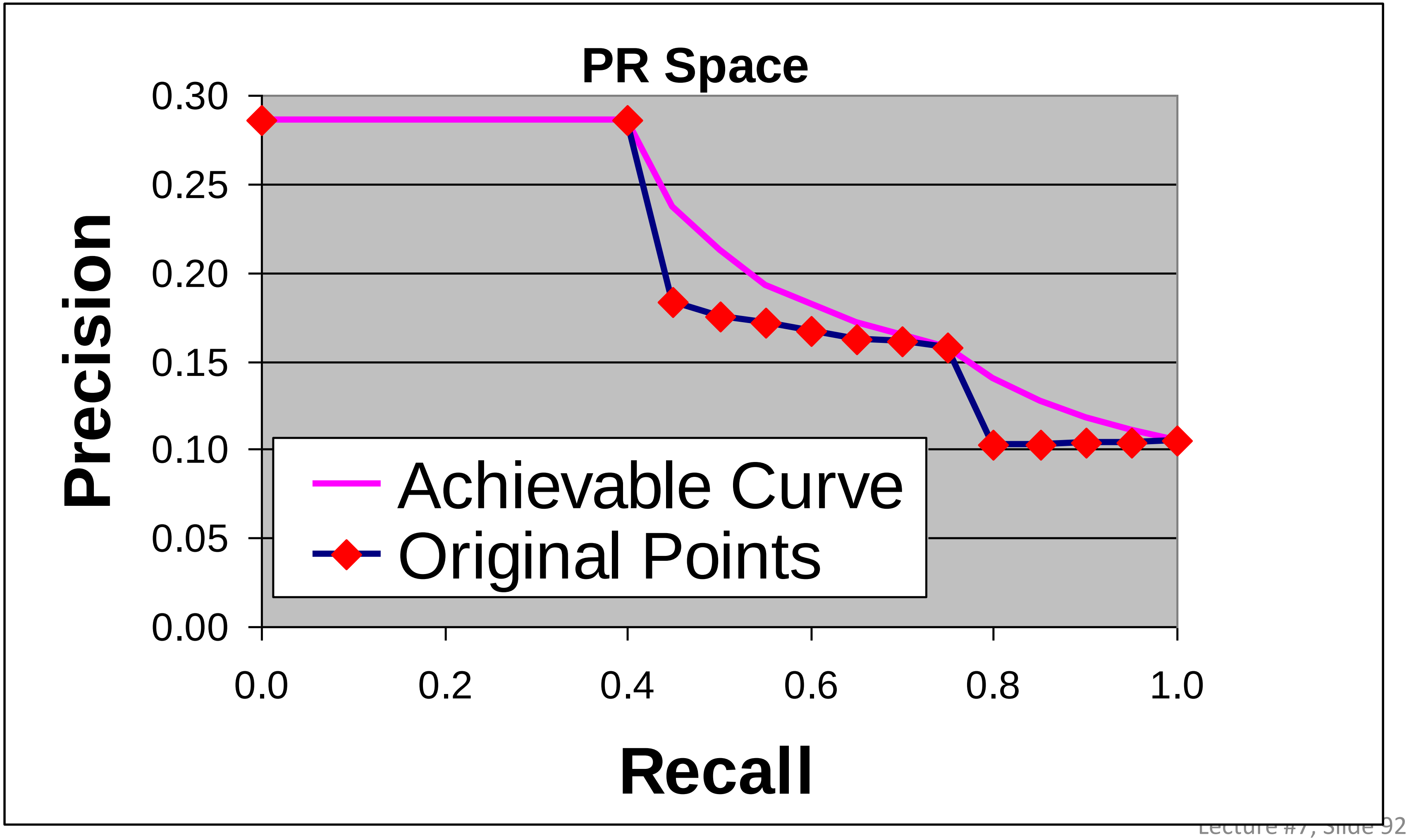
Convex Hull


ROC Space
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
Convex Hull
Original Points

A2: Achievable Curve



A2: Achievable Curve



Constructing the Achievable Curve

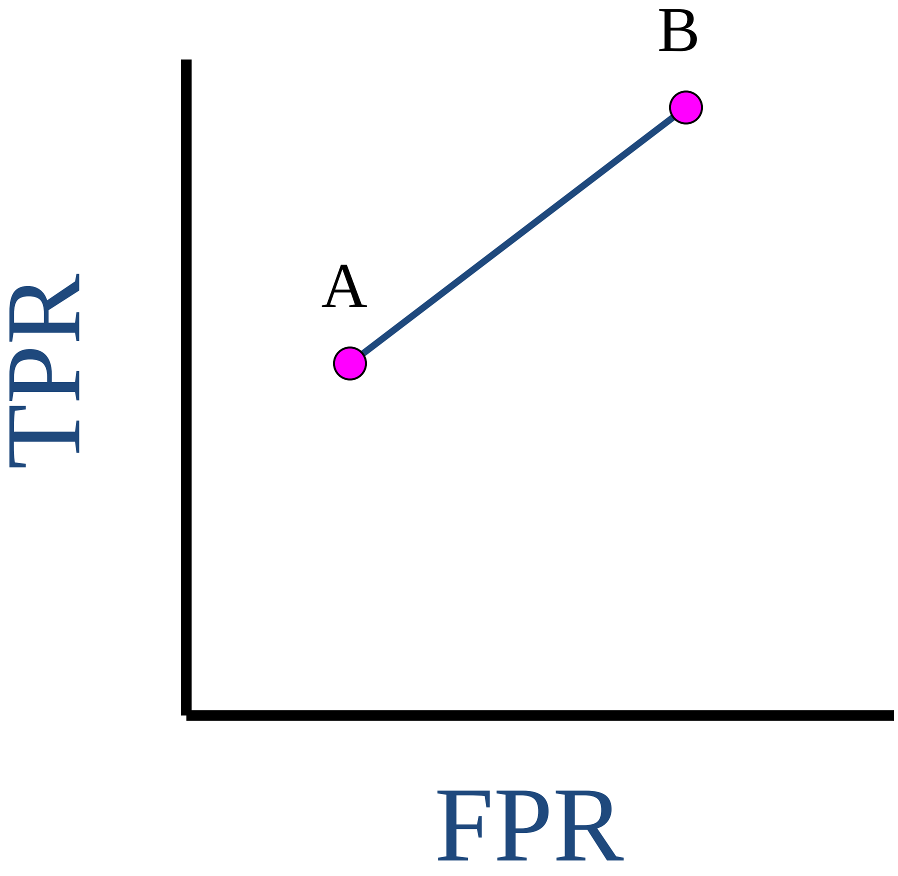
Given: Set of PR points, fixed number positive and negative examples

- Translate PR points to ROC points
- Construct convex hull in ROC space
- Convert the curve into PR space

Corollary:

By dominance theorem, the curve in PR space dominates all other legal PR curves you could construct with the given points

Q3: Interpolation



- Interpolation in ROC space is easy
- Linear connection between points

Linear Interpolation Not Achievable in PR Space

- **Precision** interpolation is counterintuitive
  [Goadrich, et al., ILP 2004]

TP	FP	TP Rate	FP Rate	Recall	Prec
500	500	0.50	0.06	0.50	0.50
750	**4750**	**0.75**	**0.53**	**0.75**	**0.14**
1000	9000	1.00	1.00	1.00	0.10

Example Counts ROC Curves PR Curves

Example Interpolation

	TP	*FP*	*REC*	*PREC*
A	5	5	0.25	0.5
B	10	30	0.5	0.25

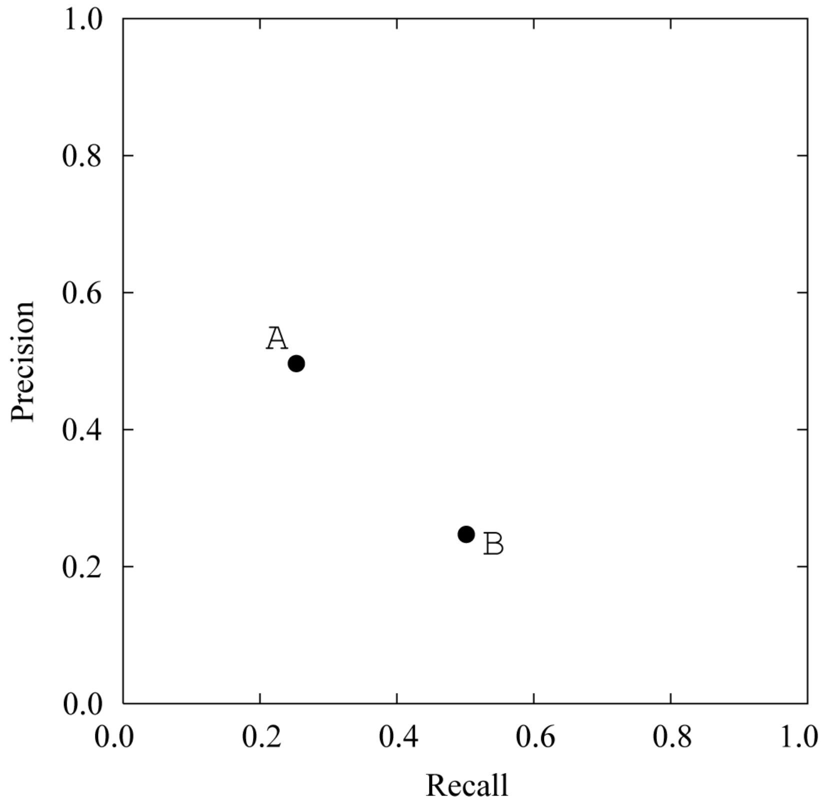
A dataset with 20 positive and 2000 negative examples

Q: For each extra TP covered, how many FPs do you cover?

A: $$\frac{FP_B - FP_A}{TP_B - TP_A}$$

Example Interpolation

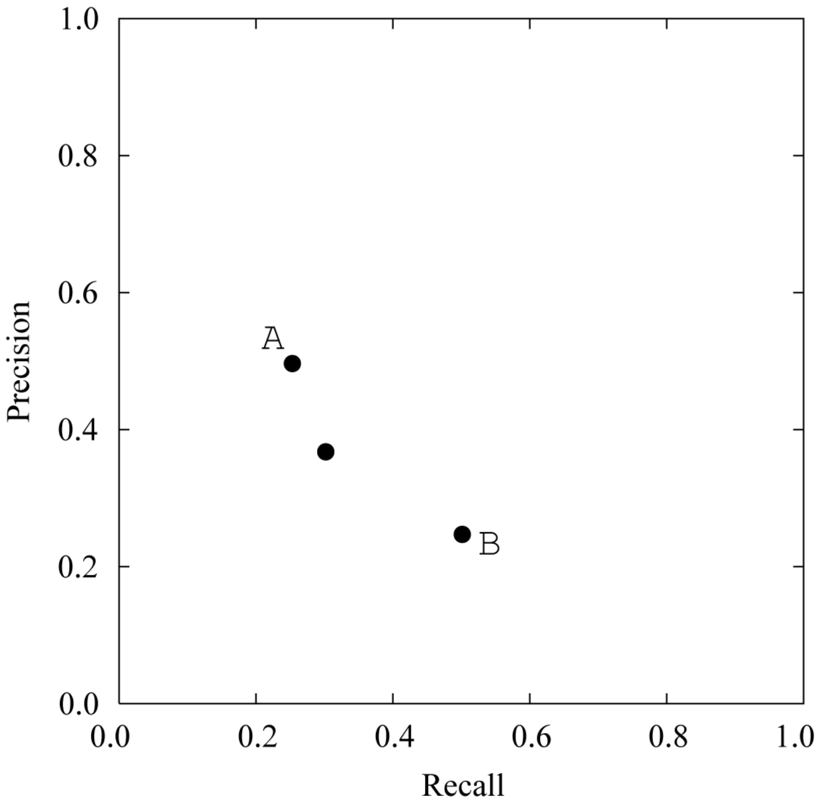
	TP	*FP*	*REC*	*PREC*
A	5	5	0.25	0.5
B	10	30	0.5	0.25



A dataset with 20 positive and 2000 negative examples

Example Interpolation

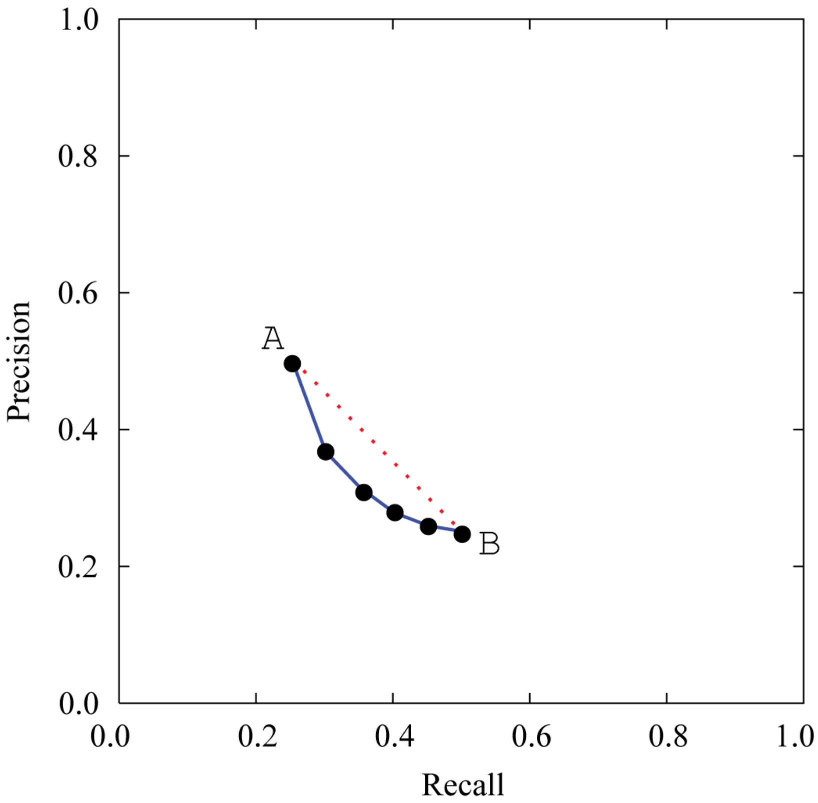
	TP	*FP*	*REC*	*PREC*
A	5	5	0.25	0.5
.	6	10	0.3	0.375
B	10	30	0.5	0.25



A dataset with 20 positive and 2000 negative examples

Example Interpolation

	TP	*FP*	*REC*	*PREC*
A	5	5	0.25	0.5
.	6	10	0.3	0.375
.	7	15	0.35	0.318
.	8	20	0.4	0.286
.	9	25	0.45	0.265
B	10	30	0.5	0.25



A dataset with 20 positive and 2000 negative examples

Optimizing AUC

- Interest in learning algorithms that optimize Area Under the Curve (AUC)
  [Ferri et al. 2002, Cortes and Mohri 2003, Joachims 2005, Prati and Flach 2005, Yan et al. 2003, Herschtal and Raskutti 2004]
- *Q: Does an algorithm that optimizes AUC-ROC also optimize AUC-PR?*
- *A: No. Can easily construct counterexample*

Outline

- Decision Trees

- Experimental Methodology
  - Methodology overview
  - How to present results
  - Hypothesis testing

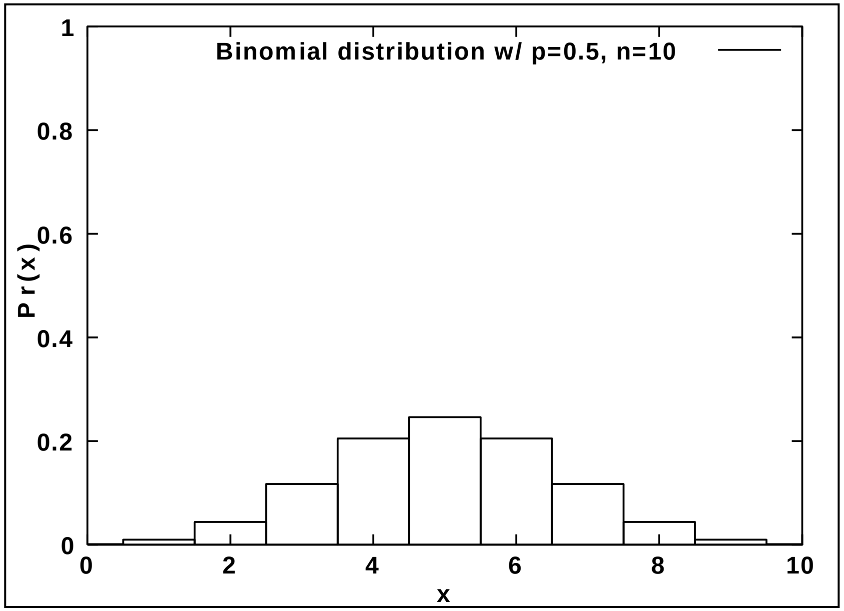
Alg 1 vs. Alg 2

- Alg 1 has accuracy 80%, Alg 2 82%
- Is this difference significant?
- Depends on how many test cases these estimates are based on
- The test we do depends on how we arrived at these estimates

The Binomial Distribution

- Distribution over the number of successes in a fixed number *n* of independent trials (with same probability of success *p* in each)

$$\Pr(x) = \binom{n}{x} p^{x} (1-p)^{n-x}$$



Leave-One-Out: Sign Test

- Suppose we ran leave-one-out cross-validation on a data set of 100 cases
- Divide the cases into (1) Alg 1 won, (2) Alg 2 won, (3) Ties (both wrong or both right); Throw out the ties
- Suppose 10 ties and 50 wins for Alg 1
- Ask: Under (null) binomial(90,0.5), what is prob of 50+ or 40- successes?

What about 10-fold?

- Difficult to get significance from sign test of 10 cases
- We're throwing out the **numbers** (accuracy estimates) for each fold, and just asking which is larger
- Use the numbers… t-test… designed to test for a difference of means

Paired Student *t*-tests

- Given
  - 10 training/test sets
  - 2 ML algorithms
  - Results of the 2 ML algo's on the 10 test-sets
- Determine
  - Which algorithm is better on this problem?
  - Is the difference statistically significant?

Paired Student *t*–Tests (cont.)

<u>Example</u>

	Accuracies on Testsets				
Algorithm 1:	80%	50	75	…	99
Algorithm 2:	79	49	74	…	98
δ_i :	+1	+1	+1	…	+1

- Algorithm 1's mean is better, but the two std. Deviations will clearly overlap
- But algorithm1 is always better than algorithm 2

δ

The Random Variable in the *t*-Test

Consider random variable

δ_i = Algo A's test-set$_i$ error minus Algo B's test-set$_i$ error

Notice we're "factoring out" test-set difficulty by looking at relative performance

In general, one tries to explain variance in results across experiments

Here we're saying that

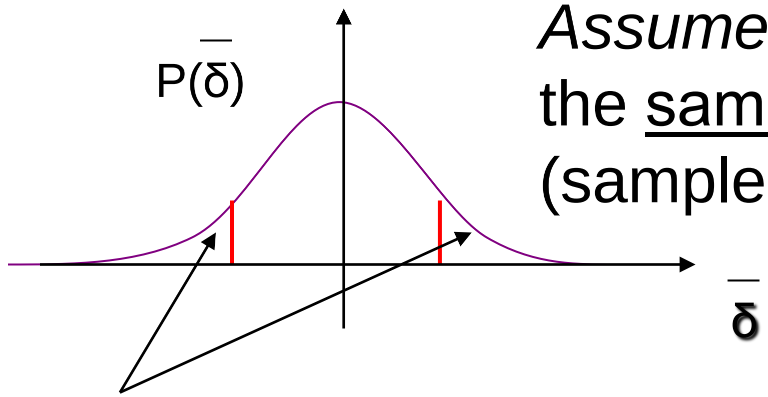
Variance = f(Problem difficulty) + g(Algorithm strength)

More on the Paired *t*-Test

- Our NULL HYPOTHESIS is that the two ML algorithms have equivalent average accuracies
  - That is, differences (in the scores) are due to the "random fluctuations" about the mean of zero
- We compute the probability that the observed δ arose from the null hypothesis
  - If this probability is low we **reject** the null hypo and say that the two algo's appear different
  - 'Low' is usually taken as prob ≤ 0.05

The Null Hypothesis Graphically

1.



½ (1 – *M*) probability mass
in each tail (ie, *M* inside)
Typically *M* = 0.95

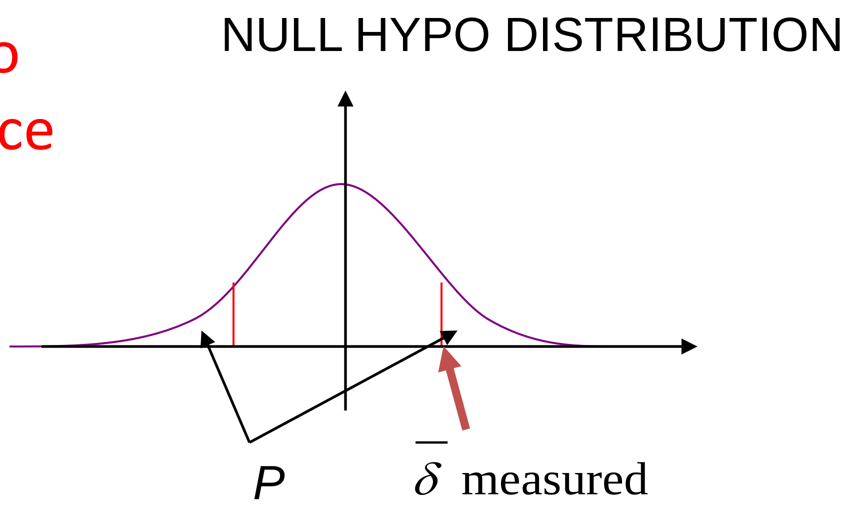
Assume <u>zero</u> mean and use the <u>sample's variance</u> (sample = experiment)

Does our measured δ lie in the regions indicated by arrows? If so, reject <u>null hypothesis,</u> since it is unlikely we'd get such a δ by chance

Some Jargon: *P*–values

P-Value = Probability of getting one's results or greater, given the NULL HYPOTHESIS

(We usually want P ≤ 0.05 to be confident that a difference is statistically significant)



"Accepting" the Null Hypothesis

Note: even if the *p*–value is high, we can<u>not</u> assume the null hypothesis is *true*

Eg, if we flip a coin twice and get one head, can we statistically infer the coin is <u>fair</u>?

Vs. if we flip a coin 100 times and observe 10 heads, we can statistically infer coin is <u>u</u>nfair because that is very unlikely to happen with a fair coin

How would we show a coin <u>*is*</u> fair?

Performing the t-Test

- Easiest way: Excel:
  - ttest(array1, array2, 2, 1)
  - Returns p-value

Assumptions of the t-Test

- Test statistical is normally distributed
  - Reasonable if we are looking at classifier accuracy
  - Not reasonable if we are looking at AUC
    - Use Wilcoxon signed-rank test

- Independent sample of test-examples
  - Violate this with 10-fold cross-validation

Next Class

- Homework 1 is due!
- Bayesian learning
  - Bayes rule
  - MAP hypothesis
- Bayesian networks
  - Representation
  - Learning
  - Inference

Summary

- Decision trees are a very effective classifier
  - Comprehensible to humans
  - Constructive, deterministic, eage
  - Make axis-parallel cuts through feature space

- Having the right experimental methodology is crucial
  - Don’t train on the test data!!
  - Many different ways to present results

end