Machine Learning: Decision Trees and Information, Evaluating ML Models

(Ch. 18.1–18.3)

1

Bookkeeping

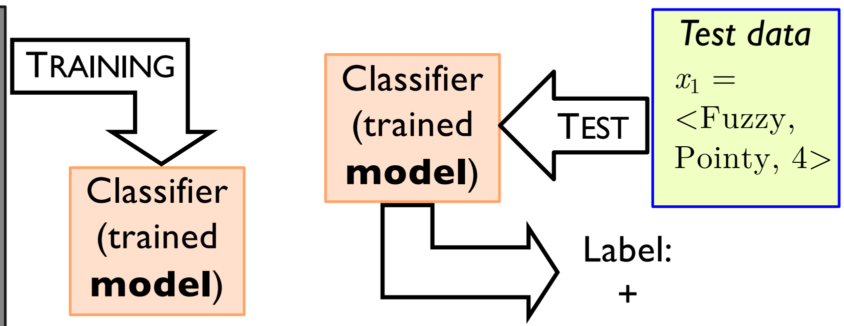
- Midterm
  - Rough curve: 60+ = A, 50+ = B, 40+ = C
  - We will go over some of the more complex questions today
  - I encourage you to go back to materials and seek answers
  - Reminder: 24 hours from exam return before we discuss grades
- HW3
  - Posted: Filtering example and spreadsheet with worked math
  - Posted: Detailed writeup on information gain
  - Nadja has office hours T and W afternoons
- Today: ML 2
  - Decision trees – entropy, information gain
  - Measuring model quality – how good is what we've learned?

2

Inductive Learning Pipeline

Training data, X

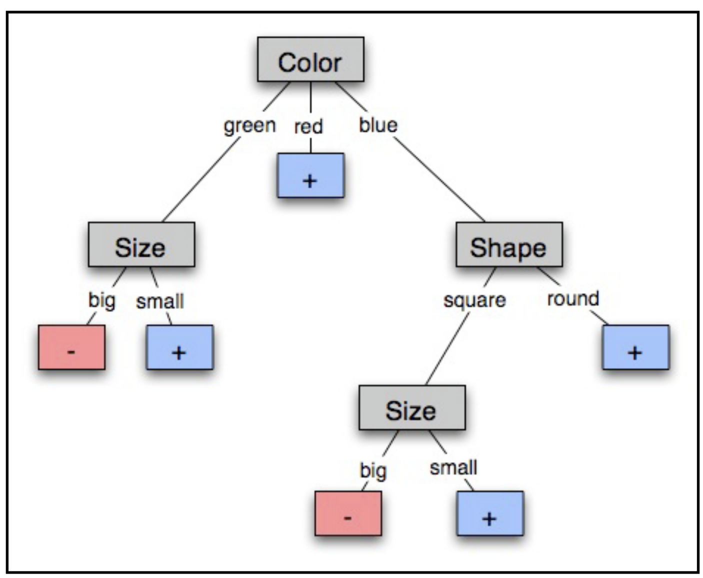
Text-ure	*Ears*	*Legs*	*Class*
Fuzzy	Round	4	+
Slimy	Missing	4	-
Fuzzy	Pointy	4	-
Fuzzy	Round	4	+
Fuzzy	Pointy	4	+
…			



3

Learning Decision Trees

- Each **non-leaf** node is an attribute (feature)
- Each **arc** is one value of the attribute at the node it comes from
- Each **leaf** node is a classification (+ or -)



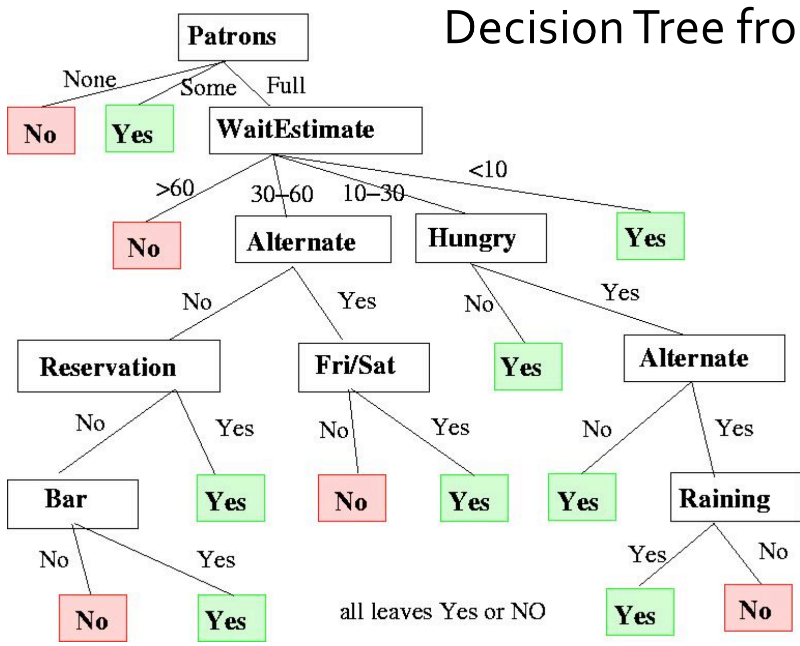
4

A Training Set

Datum	Attributes										Outcome (Label)
	altern-atives	bar	Friday	hungry	people	$	rain	reser-vation	type	wait time	Wait?
X_1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
X_2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
X_3	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
X_4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10-30	Yes
X_5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
X_7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	No
X_8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	Yes
X_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	0-30	No
X_{11}	No	No	No	No	None	$	No	No	Thai	0-10	No
X_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	Yes

5

Decision Tree from Inspection



Problem from R&N, table from Dr. Manfred Kerber @ Birmingham, with thanks – www.cs.bham.ac.uk/~mmk/Teaching/AI/l3.html

6

ID3/C4.5

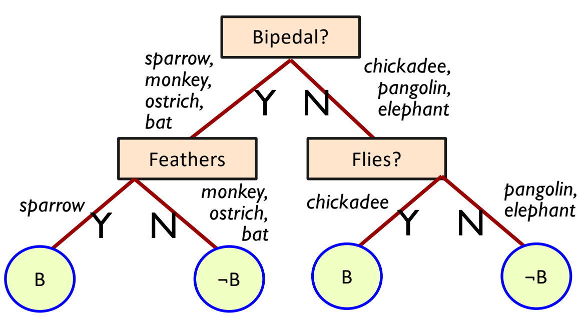
- A **greedy** algorithm for decision tree construction
  - Ross Quinlan, 1987
- Construct decision tree top-down by recursively selecting the "best attribute" to use at current node
  - Select attribute for current node
  - Generate child nodes (one for each possible value of attribute)
  - Partition training data using attribute values
  - Assign subsets of examples to the appropriate child node
  - Repeat for each child node until all examples associated with a node are either all positive or all negative

7

Bird or Not-Bird?

1.
2.
3.
4.
5.

But… we should have split on feathers first



Examples (training data)	Attributes			Outcome
	Bipedal	Flies	Feathers	
Sparrow	Y	Y	Y	B
Monkey	Y	N	N	¬B
Ostrich	Y	N	Y	B
Pangolin	N	N	N	¬B
Bat	Y	Y	N	¬B
Elephant	N	N	N	¬B
Chickadee	N	Y	Y	B

<u>Test</u>

mouse: <B:N, Fl:N, Fe:N>

8

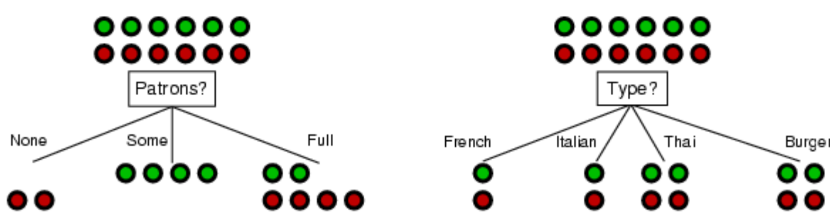
Choosing the Best Attribute

- **Key problem**: what attribute to split on?
- Some possibilities are:
  - **Random**: Select any attribute at random
  - **Least-Values**: Choose attribute with smallest number of values
  - **Most-Values**: Choose attribute with largest number of values
  - **Max-Gain**: Choose attribute that has the largest expected **information gain**—the attribute that will result in the smallest expected size of the subtrees rooted at its children
- ID3 uses Max-Gain to select the best attribute

9

Choosing an Attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative" – that is, we want *pure* groups

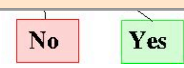


- Which is better: Patrons? or Type?
- Why?

10

ID3-induced Decision Tree

To build the best decision tree, we're going to have to talk about *information*.



11

Information Theory 101

- **Information**: the **minimum number of bits** needed to store or send some information
  - Wikipedia: "The measure of data, known as information entropy, is usually expressed by the average number of bits needed for storage or communication"
- Intuition: minimize effort to communicate/store
  - Common words (a, the, dog) are shorter than less common ones (parliamentarian, foreshadowing)
  - In Morse code, common (probable) letters have shorter encodings

"A Mathematical Theory of Communication," Bell System Technical Journal, 1948, Claude E. Shannon, Bell Labs

12

Surprisal – Information Theory 101b

- We consider events as something that does or does not happen. Each sending of a particular message is an event, E.
- The information content, or *surprisal*, of an event increases as the probability of an event decreases.
  - If you have a coin that flips heads 99% of the time, the statement “it just flipped heads” is not very informative/not very surprising
- Define the information, or surprisal, of an event E as:

  $I(E) = \log_2(1/p(E))$

  $I(E) = -\log_2(p(E))$
- Which means if the probability of an event is 1, we have 0 information.

“A Mathematical Theory of Communication,” Bell System Technical Journal, 1948, Claude E. Shannon, Bell Labs

13

Information Theory 102

- Information is (usually) measured in **bits.**
- Information in a message depends on its probability.
- Given *n* equally probable messages, what is probability p of each one?

  1/n
- Information conveyed by a message is defined as:

  $\log_2(n) = -\log_2(p)$
- Example: with 16 possible messages, $\log_2(16) = 4$, and we need 4 bits to identify/send each message

14

Entropy – Information Theory 102.b

- So, the information conveyed by a message is $\log_2(n) = -\log_2(p)$
- Given a probability distribution for *n* messages:

  $P = (p_1, p_2 \ldots p_n)$
- The information conveyed by that distribution is:

  $I(P) = -(p_1 * \log_2(p_1) + p_2 * \log_2(p_2) + .. + p_n * \log_2(p_n)) = -\Sigma_i(p_i \log_2(p_i))$
- This is the **entropy** of P: the average number of bits (per message) needed to represent a stream of messages
  - Note that we sometimes use S for entropy.

15

Information Theory 103

- Entropy: **average** number of bits (per message) needed to represent a stream of messages

  $I(P) = -(p_1 * \log_2(p_1) + p_2 * \log_2(p_2) + .. + p_n * \log_2(p_n)) = -\Sigma_i(p_i \log_2(p_i))$
- Examples (datasets resulting from flipping biased coins):
  - **P = (0.5, 0.5)**; $I(P) = -(0.5 * \log_2(0.5) + (0.5 * \log_2(0.5)) = \mathbf{1}$ → entropy of a fair coin flip
  - **P = (0.67, 0.33)**; $I(P) = -(0.67 * \log_2(0.67) + (0.33 * \log_2(0.33)) = \mathbf{0.92}$
  - **P = (0.99, 0.01)**; $I(P) = -(0.99 * \log_2(0.99) + (0.01 * \log_2(0.01)) = \mathbf{0.08}$
  - **P = (1, 0)**; $I(P) = -(1 * \log_2(1) + (0 * \log_2(0)) = \mathbf{0}$
- **As the distribution becomes more skewed, the amount of information needed to tell me what happened *decreases.* Why?**
- **Because I can just predict the most likely element, and usually be right**

16

Information Theory 103b

- Entropy over a dataset
- Consider a dataset with 1 blue, 2 greens, and 3 reds: ••••••
- I(••••••) $= -\Sigma_i \, (p_i \log_2(p_i))$

$$= -(p_b \log_2(p_b) + (p_g \log_2(p_g)) + (p_r \log_2(p_r))$$

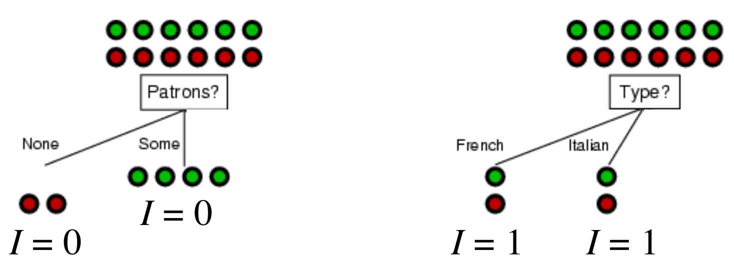
$$= -(\tfrac{1}{6} \log_2(\tfrac{1}{6}) + (\tfrac{1}{3} \log_2(\tfrac{1}{3})) + (\tfrac{1}{2} \log_2(\tfrac{1}{2}))$$

$$= 1.46$$

17

Entropy Interlude

- Entropy (I): the homogeneity of a sample
  - If everything is the same, $S = 0$
  - If differences are even $S = 1$



- So a dataset consisting entirely of people who choose to wait for a table has entropy 0, because it's very pure (homogeneous)
- A dataset of half-waiting, half-leaving has entropy of 1

18

Entropy as Measure of **Homogeneity of Examples**

- Entropy can be used to characterize the (im)purity of an arbitrary collection of examples
- **Low entropy** implies **high homogeneity (purity)**
  - Given a collection *S* (like the table of 12 examples for the restaurant domain), containing positive and negative examples of some target concept, the entropy of *S* relative to its Boolean classification is:

$$I(S) = -(p_+ * \log_2 (p_+) + p_- * \log_2 (p_-))$$

Entropy([7+, 7-]) = $-(0.5 * \log_2(0.5) + (0.5 * \log_2(0.5))$ **= 1**

Entropy([9+, 5-]) = $-(9/14 * \log_2(9/14) + (5/14 * \log_2(5/14))$

= $-(0.64 * \log_2(0.64) + (0.357 * \log_2(0.357))$ **= 0.940**

19

Information Gain

- **Information gain**: how much entropy decreases (homogeneity increases) when a dataset is split on an attribute.
  - High homogeneity → high likelihood samples will have the same class
- This is what we want! A decision tree in which we efficiently split on attributes in order to reach sets of data with homogeneous decisions
  - That is, we compare the entropy of the dataset(s) before and after the split
- Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches)

20

Information Gain, cont.

- Use to rank attributes and build decision tree!
- Choose nodes using attribute with **greatest information gain**
  - → means least information remaining after split
  - I.e., subsets are all as skewed as possible
- Why?
  - Create small decision trees: predictions can be made with few attribute tests
  - Try to find a minimal process that still captures the data (Occam's Razor)

21

Information Gain: Using Information

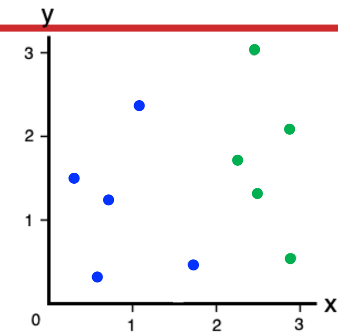
- A chosen attribute A divides the training set S into subsets $S_1, \ldots, S_v$ according to their values for A, where A has *v* distinct values.
- The information gain IG(S,A) (or just IG(S)) of an attribute A relative to a collection of examples S is defined as:

$$IG(S, A) = I(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times I(S_v)$$

- This is the gain in information due to attribute A
  - Expected reduction in entropy
- This represents the difference between
  - I(S) – the entropy of the original collection S
  - Remainder(A) - expected value of the entropy after S is partitioned using attribute A

23

Information Gain: Example



- First we calculate the entropy *before* the split, $I(S)$
  - $I(\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet) = \mathbf{1}$ (perfectly balanced)
- Split, then calculate the entropy of each branch
  - $I_{left}(\bullet\bullet\bullet\bullet) = \mathbf{0}$ (pure)
  - $I_{right}(\bullet\bullet\bullet\bullet\bullet\bullet) = -(\frac{1}{6}\log_2(\frac{1}{6}) + \frac{5}{6}\log_2(\frac{5}{6})) = \mathbf{0.65}$
- Then we calculate the entropy of the split by weighting each branch's entropy by how many data points that branch covers
  - *Left* has 4 data points: 4/10 of the data, 0.4. *Right* has 0.6 of the data.
  - $I_{split} = (0.4*0) + (0.6*0.65) = \mathbf{0.39}$
- Information gain = $\mathbf{1 - 0.39 = 0.61}$

$$IG(S,A) = I(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \times I(S_v)$$

example from victorzhou.com/blog/information-gain/

How Well Does it Work?

- At least as accurate as human experts (sometimes)
  - Diagnosing breast cancer: humans correct 65% of the time; decision tree classified 72% correct
  - BP designed a decision tree for gas-oil separation for offshore oil platforms; replaced an earlier rule-based expert system
  - Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example
  - SKICAT (Sky Image Cataloging and Analysis Tool) used a DT to classify sky objects **an order of magnitude** fainter than was previously possible, with an accuracy of over 90%.

Extensions of the Decision Tree Learning Algorithm

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is a (more applicable) extension of ID3 that accounts for real-world problems: unavailable values, continuous attributes, pruning decision trees, rule derivation, …

27

Using Gain Ratios

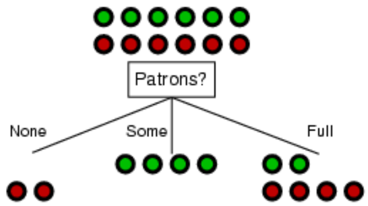
- Information Gain can be biased towards attributes A with many values v
  - Tiny subsets tend to be pure – not because they're good, just because they're small
    - Degenerate case: If attribute A has a distinct value for each record, then Info(A,S) is 0, so Gain(A,S) is maximal
  - This can give trees that generalize poorly
- To compensate for this Quinlan suggests using the following ratio instead:
  - **GainRatio(A,S) = Gain(A,S) / SplitInfo(A,S)**
  - SplitInfo: A number that's big when there are many small subsets
- SplitInfo(A,S) is the information due to the split of S on the basis of attribute A
  - SplitInfo(D,T) = $I(|S_1|/|S|, |S_2|/|S|, .., |S_v|/|S|) = - \Sigma_{v \in values(A)} |S_v|/|S| \log_2 |S_v|/|S|$
  - where $\{S_1, S_2, .. S_v\}$ is the partition of S induced by value of A

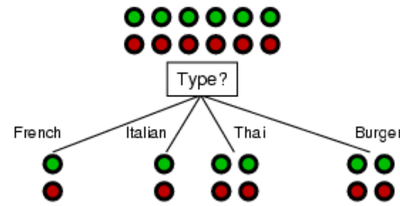
I like this short video: www.youtube.com/watch?v=rb1jdBPKzDk

28

Computing Gain Ratio

- I(S) = 1
- I(Pat, S) = .47
- I(Type, S) = 1





Gain (Pat, S) =.53
Gain (Type, S) = 0

SplitInfo (Pat, S) = ?

SplitInfo (Type, S) = ?

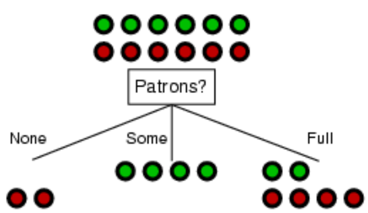
GainRatio (Pat, S) = Gain (Pat, S) / SplitInfo(Pat, S) = .53 / ______ = ?

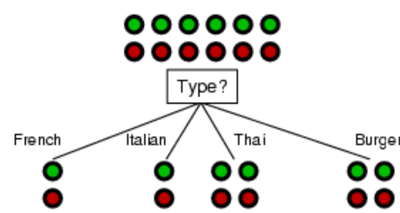
GainRatio (Type, S) = Gain (Type, S) / SplitInfo (Type, S) = 0 / ______ = ?

29

Computing Gain Ratio

- I(S) = 1
- I(Pat, S) = .47
- I(Type, S) = 1





Gain (Pat, S) =.53
Gain (Type, S) = 0

SplitInfo (Pat, S) = $- (\underline{\tfrac{1}{6} \log \tfrac{1}{6}} + \underline{\tfrac{1}{3} \log \tfrac{1}{3}} + \underline{\tfrac{1}{2} \log \tfrac{1}{2}})$
$= \tfrac{1}{6} * 2.6 + \tfrac{1}{3} * 1.6 + \tfrac{1}{2} * 1$
= **1.47**

SplitInfo (Type, S) = ?

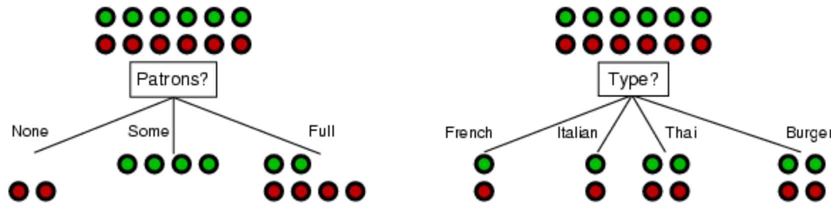
GainRatio (Pat, S) = Gain (Pat, S) / SplitInfo(Pat, S) = .53 / ______ = ?

GainRatio (Type, S) = Gain (Type, S) / SplitInfo (Type, S) = 0 / ______ = ?

30

Computing Gain Ratio

- I(S) = 1
- I(Pat, S) = .47
- I(Type, S) = 1



Gain (Pat, S) =.53
Gain (Type, S) = 0

SplitInfo (Pat, S) = - ($\frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{2}\log\frac{1}{2}$)
= $\frac{1}{6}$ *2.6 + $\frac{1}{3}$ *1.6 + $\frac{1}{2}$ *1
= **1.47**

SplitInfo (Type, S) = $\frac{1}{6}\log\frac{1}{6} + \frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{3}\log\frac{1}{3}$
= $\frac{1}{6}$ *2.6 + $\frac{1}{6}$ *2.6 + $\frac{1}{3}$ *1.6 + $\frac{1}{3}$ *1.6 = **1.93**

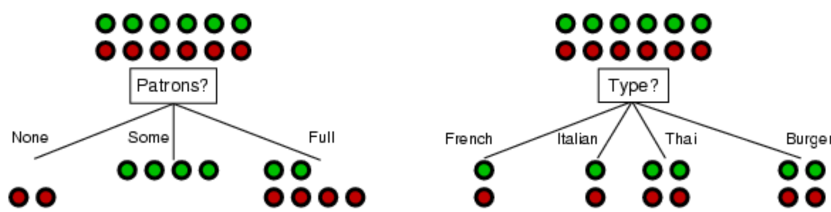
GainRatio (Pat, S) = Gain (Pat, S) / SplitInfo(Pat, S) = .53 / ______ = ?

GainRatio (Type, S) = Gain (Type, S) / SplitInfo (Type, S) = 0 / ______ = ?

31

Computing Gain Ratio

- I(S) = 1
- I(Pat, S) = .47
- I(Type, S) = 1



Gain (Pat, S) =.53
Gain (Type, S) = 0

SplitInfo (Pat, S) = - ($\frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{2}\log\frac{1}{2}$)
= $\frac{1}{6}$ *2.6 + $\frac{1}{3}$ *1.6 + $\frac{1}{2}$ *1
= 1.47

SplitInfo (Type, S) = $\frac{1}{6}\log\frac{1}{6} + \frac{1}{6}\log\frac{1}{6} + \frac{1}{3}\log\frac{1}{3} + \frac{1}{3}\log\frac{1}{3}$
= $\frac{1}{6}$ *2.6 + $\frac{1}{6}$ *2.6 + $\frac{1}{3}$ *1.6 + $\frac{1}{3}$ *1.6 = 1.93

GainRatio (Pat, S) = Gain (Pat, S) / SplitInfo(Pat, S) = .53 / 1.47 = .36

GainRatio (Type, S) = Gain (Type, S) / SplitInfo (Type, S) = 0 / 1.93 = 0

32

Real-Valued Data

- Select thresholds defining intervals so each becomes a discrete value of attribute
- Use heuristics, e.g. always divide into quartiles
- Use domain knowledge, e.g. divide age into infant (0-2), toddler (3-5), school-aged (5-8)
- Or treat this as another learning problem
  - Try different ways to discretize continuous variable; see which yield better results w.r.t. some metric
  - E.g., try midpoint between every pair of values

33

Converting Decision Trees to Rules

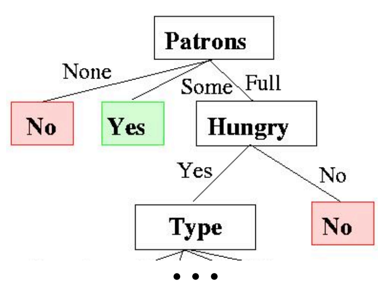
- 1 rule for each path in tree (from root to a leaf)
- Left-hand side: labels of nodes and arcs
- Right-hand side: classification

  Patrons=None → Don't wait

  Patrons=Some → Wait

  Patrons=Full ∧ Hungry=No → Don't wait

  etc…



- Resulting rules can be simplified and reasoned over

34

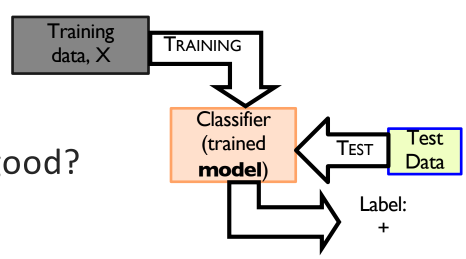
Summary: Decision Tree Learning

- (Still!) one of the most widely used learning methods in practice
- Can out-perform human experts in many problems
  - Strengths:
    - Fast
    - Simple to implement
    - Can convert to a set of easily interpretable rules
    - Empirically valid in many commercial products
    - Handles noisy data
  - Weaknesses:
    - Univariate splits/Partitioning using only one attribute at a time (limits types of possible trees)
    - Large trees hard to understand
    - Requires fixed-length feature vectors
    - Non-incremental (i.e., batch method)

36

ML: Measuring Model Quality

- So we have training data, and we have learned a model
  - A learned decision tree is one such model
- We have some set of test data we have held out
- How do we evaluate whether the model is good?
- How can this process fail?



37

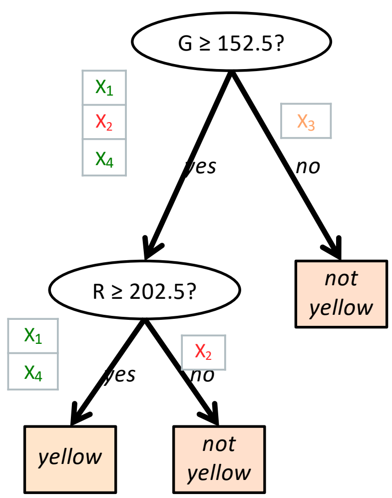
Measuring Model Quality

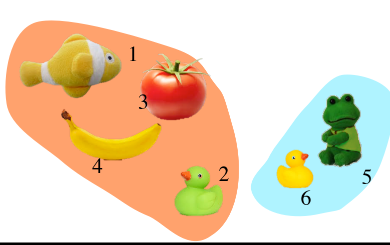
- How good is a model?
  - Predictive accuracy
  - False positives / false negatives for a given cutoff threshold
    - Loss function (accounts for cost of different types of errors)
  - Area under the curve
  - Minimizing loss can lead to problems with overfitting

38

One Possible Decision Tree

sample	*attributes*				*label*
	R	*G*	*B*	*Fuzzy?*	*Yellow?*
X_1	205	200	40	Y	yes
X_2	90	250	90	N	no
X_3	220	10	22	N	no
X_4	205	210	10	N	yes



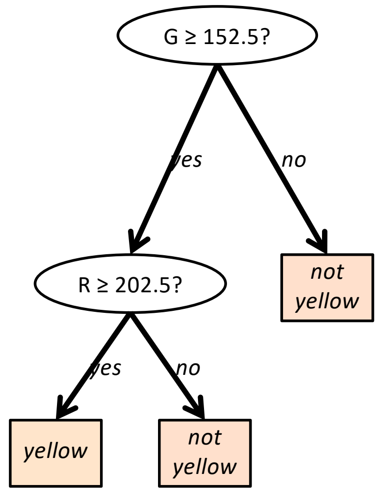


39

One Possible Decision Tree

- Predictions

	R	G	B	Fuzzy?	Prediction: Is it yellow?
X_7	215	45	190	N	



40

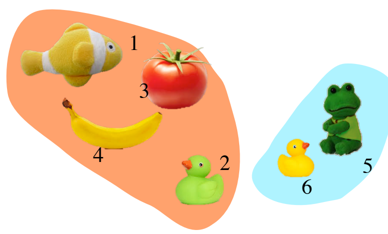
Measuring Model Quality

- Training error
  - Train on all data; measure error on all data
  - Subject to overfitting (of course we'll make good predictions on the data on which we trained!)
- Regularization
  - Attempt to avoid overfitting
  - Explicitly minimize the complexity of the function while minimizing loss
  - Tradeoff is modeled with a regularization parameter

41

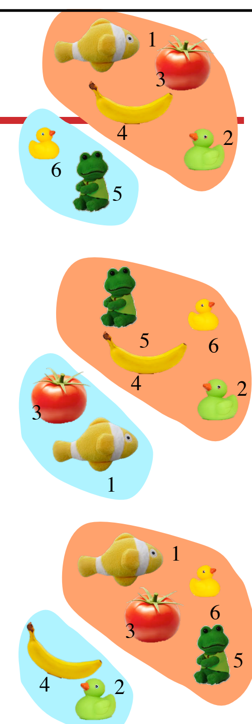
Cross-Validation

- Holdout cross-validation:
  - Divide data into training set and test set
  - Train on training set; measure error on test set
  - Better than training error, since we are measuring generalization to new data
  - To get a good estimate, we need a reasonably large test set
  - But this gives less data to train on, reducing our model quality!



42

Cross-Validation, cont.



- *k*-fold cross-validation:
  - Divide data into *k* folds
  - Train on *k*-1 folds, use the k^{th} fold to measure error
  - Repeat *k* times; use average error to measure generalization accuracy
  - Statistically valid and gives good accuracy estimates
  - 5 and 10 are common values for *k*
- Leave-one-out cross-validation (LOOCV)
  - *k*-fold cross validation where *k*=N (test data = 1 instance!)
  - Quite accurate, but also quite expensive, since it requires building N models

43

Correctness

- True positive
  - I predict it's yellow, and it is yellow
- True negative
  - I predict it's not yellow, and it's not
- False positive
  - I predict it's yellow, but it's not
- False negative
  - I predict it's not yellow, but it is

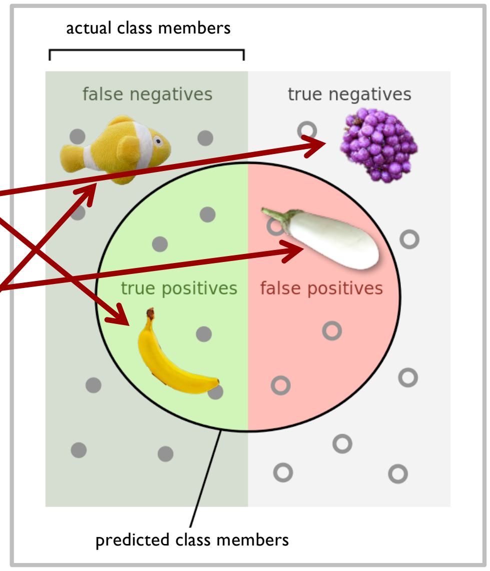


Figure: en.wikipedia.org/wiki/Precision_and_recall

44

On Sensitivity and Specificity

- Sensitivity (recall) measures avoidance of false negatives
- Specificity (precision) measures avoidance of false positives
- TSA security scenario:
  - Metal scanners set for low specificity (e.g., trigger on keys) to reduce risk of missing dangerous objects
  - Result is high sensitivity overall
- Cancer test scenario:
  - Screening exam given to lots of people: also high sensitivity (better to flag someone for followup testing incorrectly, than to miss someone)
  - Detail exam: need high specificity

45

Precision/Recall

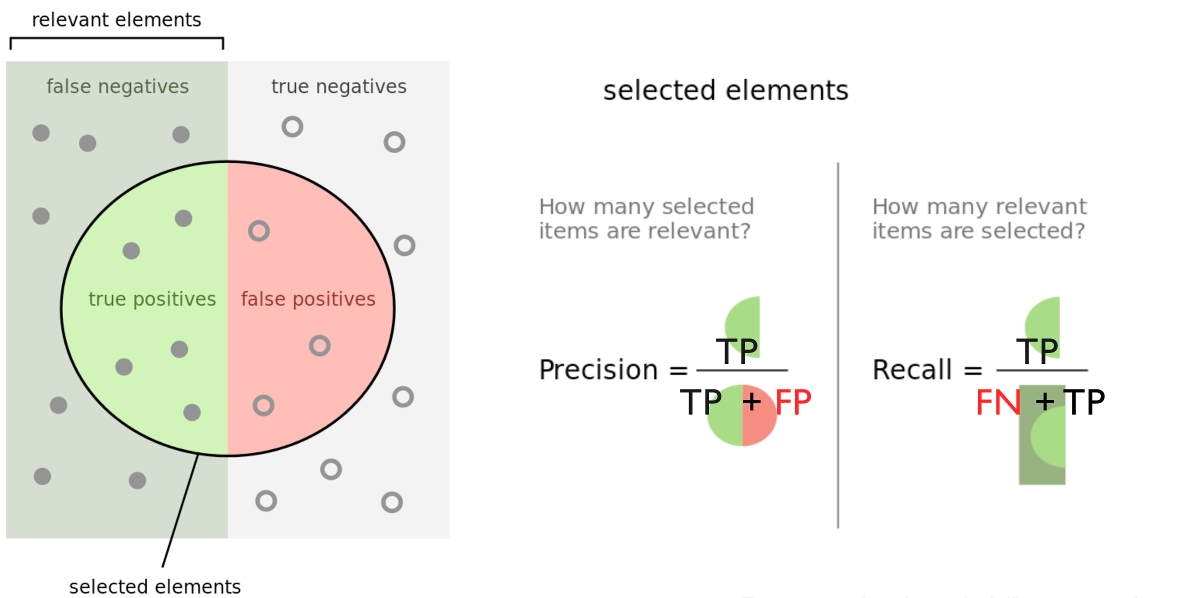
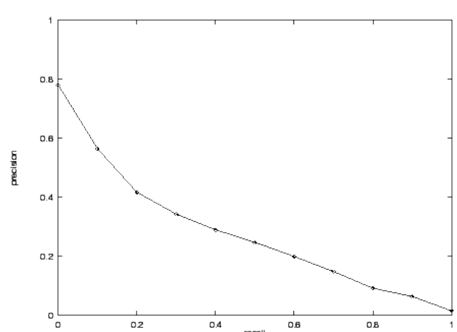


Figure: en.wikipedia.org/wiki/Precision_and_recall

46

Precision, or Recall?

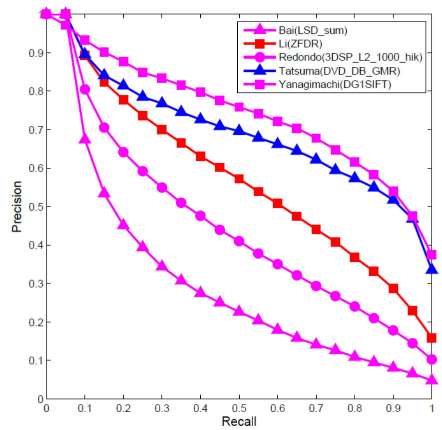
- Precision (specificity) and recall (sensitivity) are in tension
- In general, increasing one causes the other to decrease
  - The more *precise* you are, the more things you will miss
  - The more you guarantee you will catch everything, the more you will return some incorrect things (casting a wide net)
- So… which is better?
  - Recall our cancer example
- Studying the precision/recall curve is informative



47

Precision and Recall

- If one system's curve is always above the other, it's better



48

F measure

- The F1 measure combines both into a useful single metric

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$= \frac{TP}{TP + 1/2\ (FP + FN)}$$

- Idea: both precision and recall need to be reasonably good
- Heavily penalizes small precision or small recall
- Can be tuned with different values for F to prefer recall or precision

49

Confusion Matrix (1)

- A confusion matrix can be a better way to show results
- For binary classifiers it's simple and is related to type I *and* type II errors (i.e., false positives and false negatives)
- There may be different costs for each kind of error
- So we need to understand their frequencies

predicted (columns); actual (rows)

a/c	C	~C
C	True positive	False negative
~C	False positive	True negative

56

Confusion Matrix (2)

- For multi-way classifiers, a confusion matrix is even more useful
- It lets you focus in on where the errors are

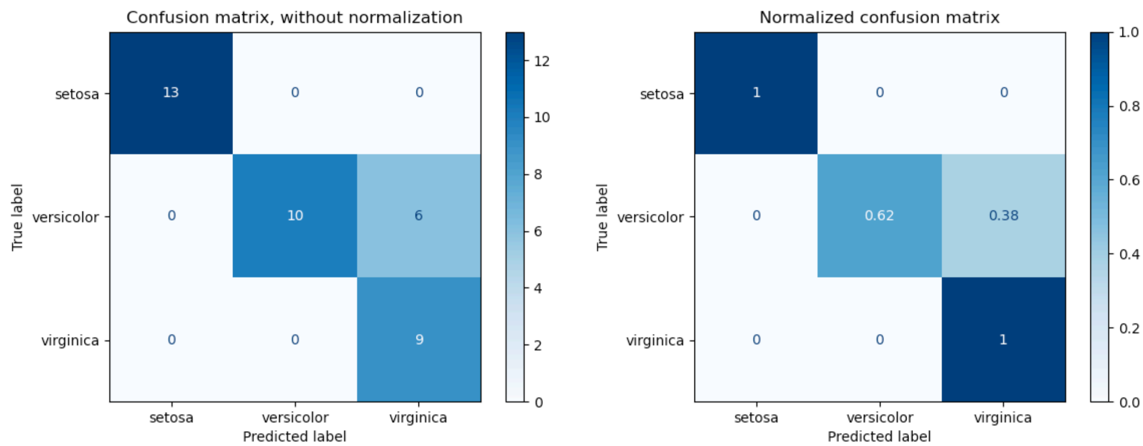
predicted (columns); actual (rows)

	Cat	Dog	rabbit
Cat	5	3	0
Dog	2	3	1
Rabbit	0	2	11

57

Confusion Matrix (2)

- For multi-way classifiers, a confusion matrix is even more useful
- It lets you focus in on where the errors are



Figures: scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

58

Overfitting

- Sometimes, model fits training data well but doesn't do well on test data
- Can be it "overfit" to the training data
  - Model is too specific to training data
  - Doesn't generalize to new information well
- Learned model: (Y∧Y∧Y→B ∨ Y∧N∧N→M ∨ ...)

Examples (training data)	Attributes			Outcome
	Bipedal	Flies	Feathers	
Sparrow	Y	Y	Y	B
Monkey	Y	N	N	M
Ostrich	Y	N	Y	B
Bat	Y	Y	N	M
Elephant	N	N	N	M

59

Overfitting 2

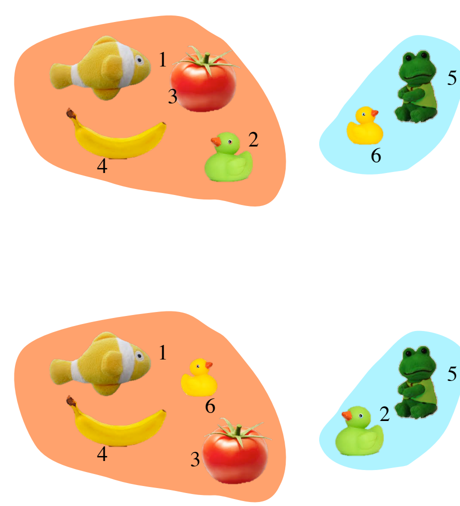
- Irrelevant attributes → overfitting
- If hypothesis space has many dimensions (many attributes), may find meaningless regularity
  - Ex: Name starts with [A-M] → Mammal
  - Problem is that we have a feature that doesn't really pertain to the classification problem

Examples (training data)	Attributes			Outcome
	Bipedal	Flies	Feathers	
Sparrow	Y	Y	Y	B
Monkey	Y	N	N	M
Ostrich	Y	N	Y	B
Bat	Y	Y	N	M
Elephant	N	N	N	M

60

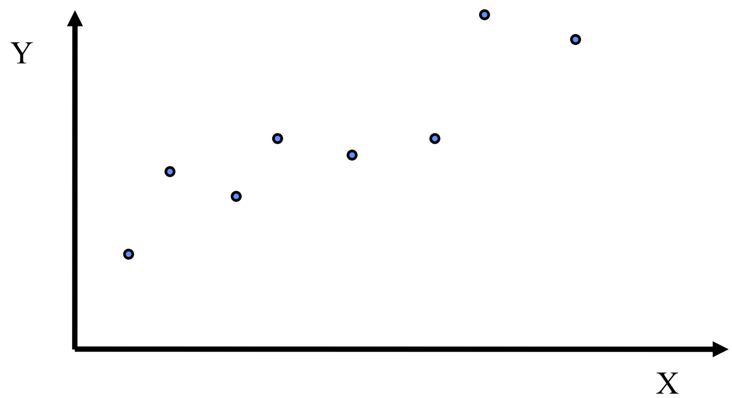
Sources of Overfitting

- Incomplete training data
  - Including small training data
- Bad training/test split
- Irrelevant attributes in feature set
- "Overtraining"
  - Sometimes it makes sense to stop before training has learned all it can
- Poor choice of model/ML algorithm



61

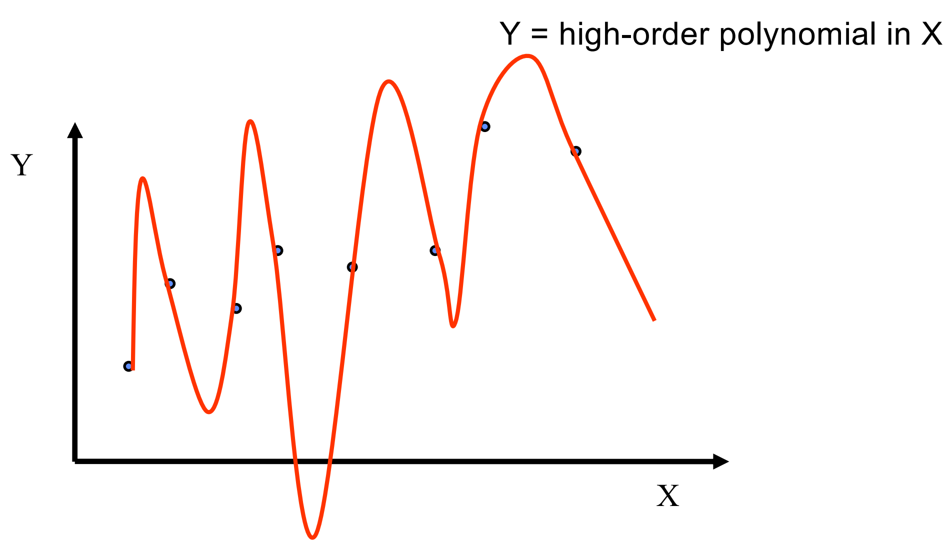
Overfitting and Underfitting



Slide credit Richard H. Lathrop

62

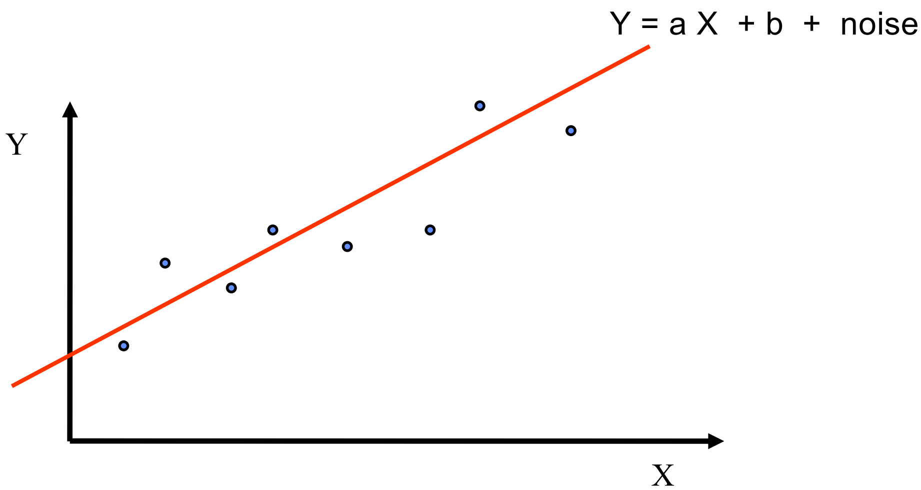
A Complex Model



Slide credit Richard H. Lathrop

63

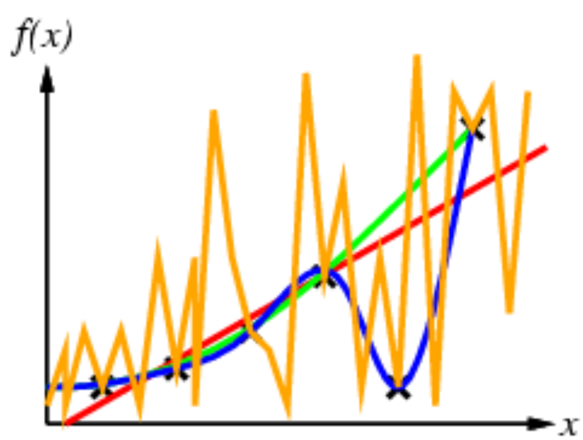
A Much Simpler Model



Slide credit Richard H. Lathrop

64

Another example



- What you choose says a lot about what kind of learning you're doing; for example, the green line omits outliers, suggesting you suspect noisy data

Slide credit Richard H. Lathrop

65

Overfitting

- Fix by…
  - Getting more training data (an ML panacea)
  - Removing irrelevant features (e.g., remove 'first letter' from bird/mammal feature vector)
  - In decision trees, pruning low nodes (e.g., if improvement from best attribute at a node is below a threshold, stop and make this node a leaf rather than generating child nodes)
- Regularization
- Lots of other choices…

66

Noisy Data

- Many kinds of "noise" can occur in the examples:
  - Two examples have same attribute/value pairs, but different classifications
  - Some values of attributes are incorrect
    - Errors in the data acquisition process, the preprocessing phase, …
  - Classification is wrong (e.g., + instead of -) because of some error
  - Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome
  - Some attributes are missing (are pangolins bipedal?)

67

Summary of Model Evaluation

- Data can be noisy, models can be wrong
- We can evaluate how good a model is with precision, recall, and F1
- We can visualize model results with confusion matrices
- Cross-validation lets us get more statistical power from our training data while still giving meaningful test results
- Overfitting remains a significant problem

- Questions before we do some midterm problems?

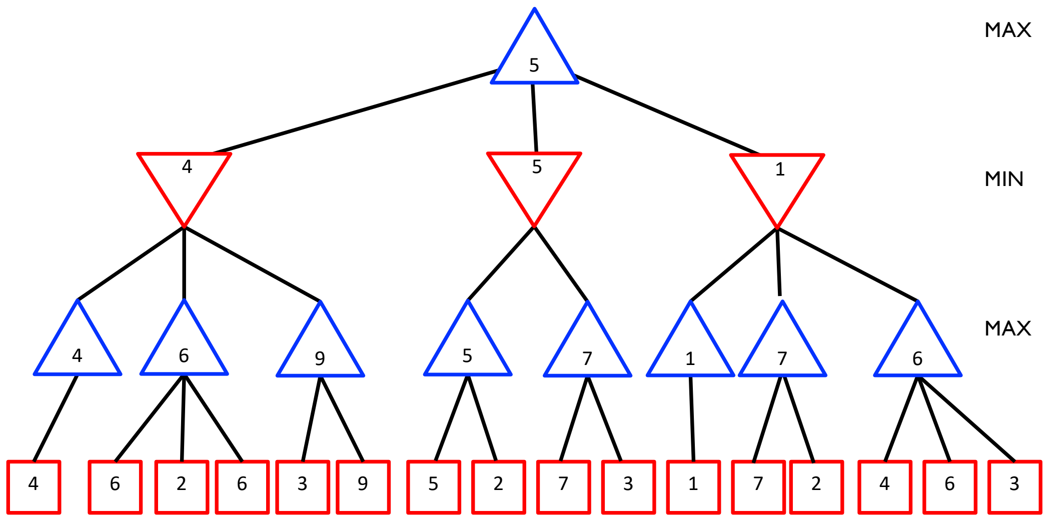
68

Some notes from the Fall 22 midterm

- Alpha-beta pruning
- Expectiminimax trees
- Constraint satisfaction
- Belief net calculations
- Admissible heuristics
- Iterative deepening
- Game theory

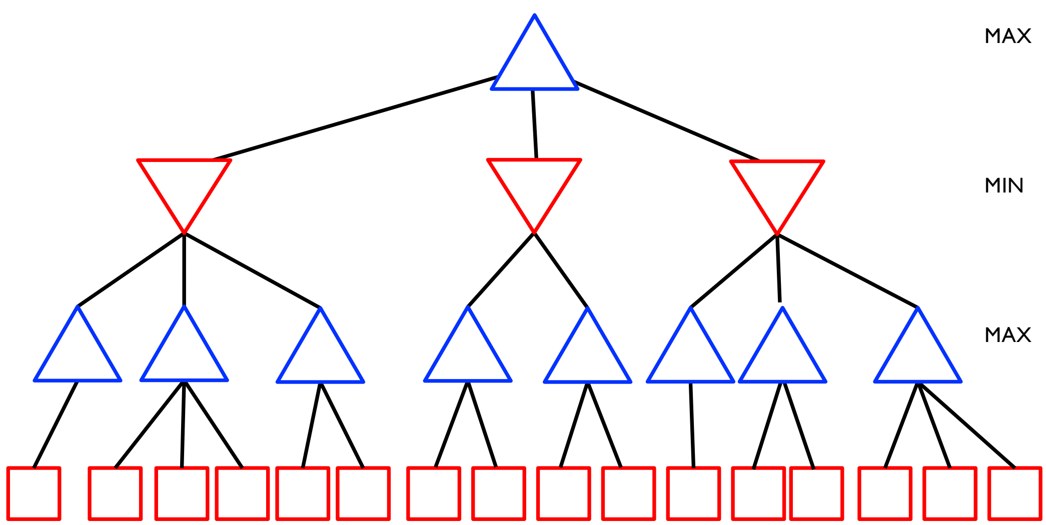
69

Alpha-beta pruning



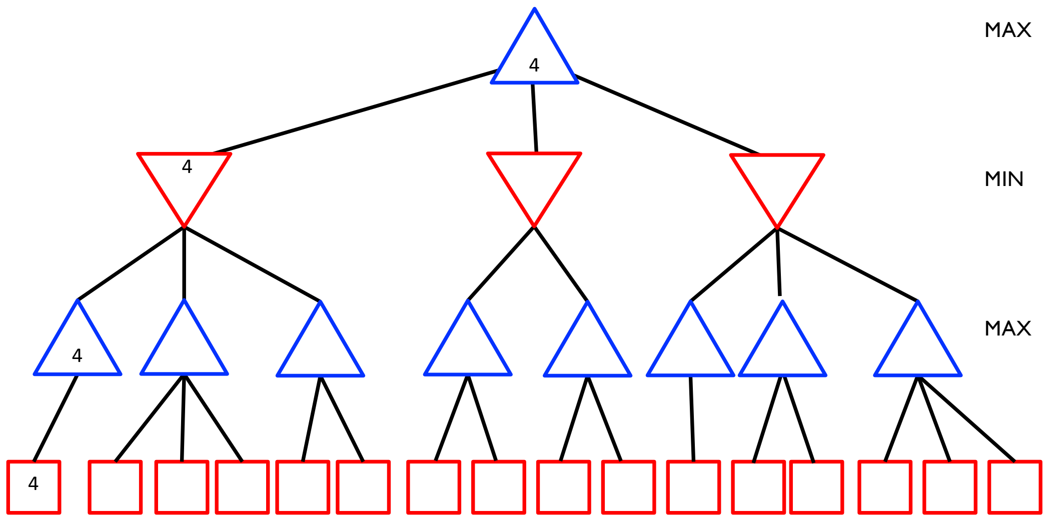
70

Alpha-beta pruning



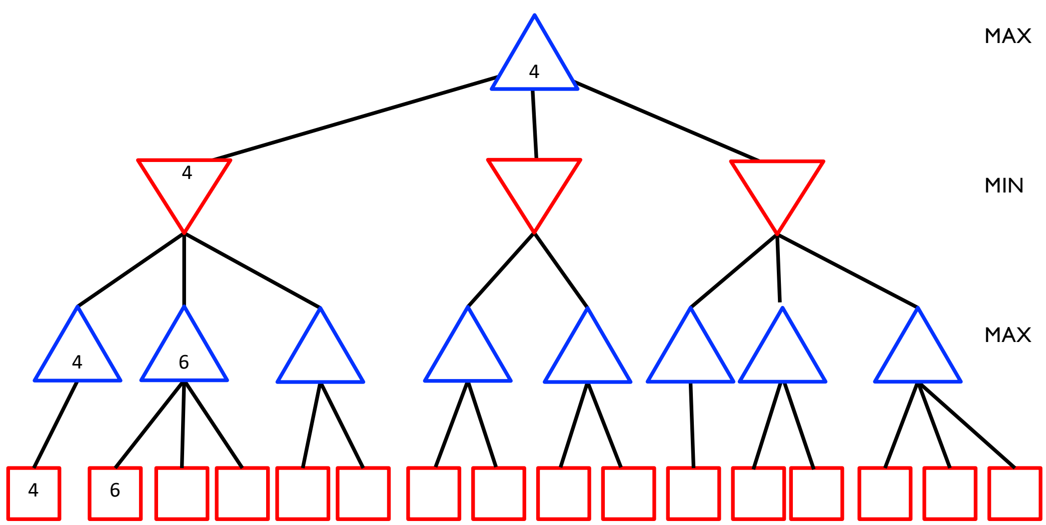
71

Alpha-beta pruning



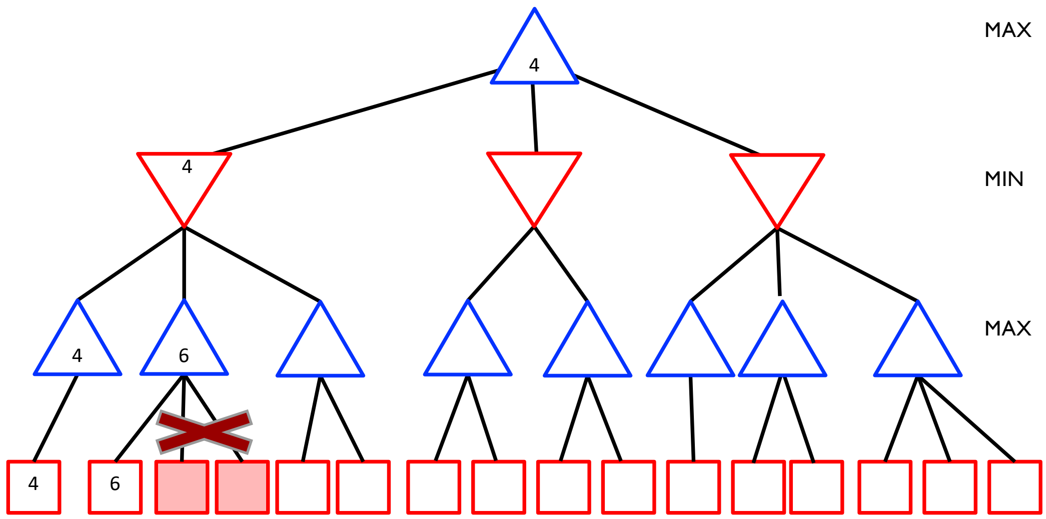
72

Alpha-beta pruning



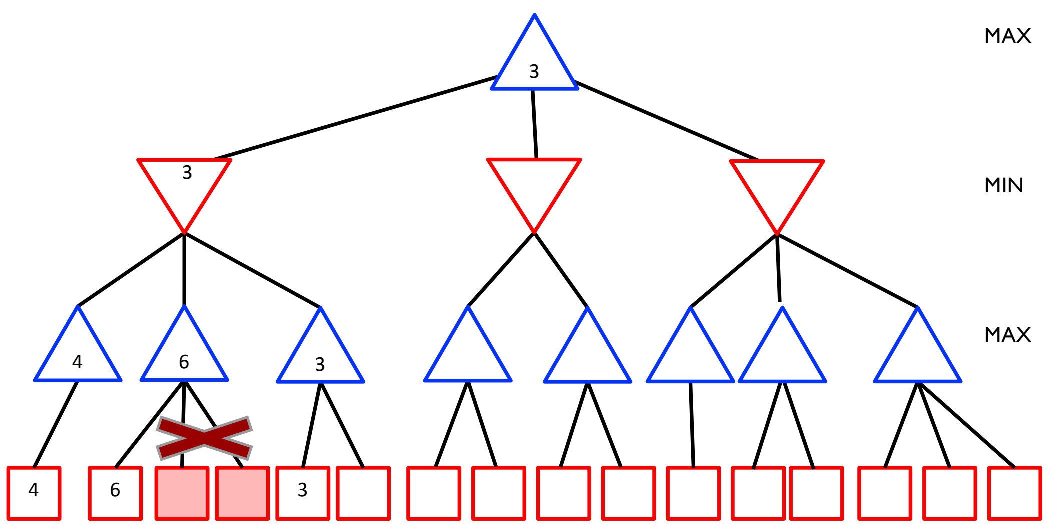
73

Alpha-beta pruning



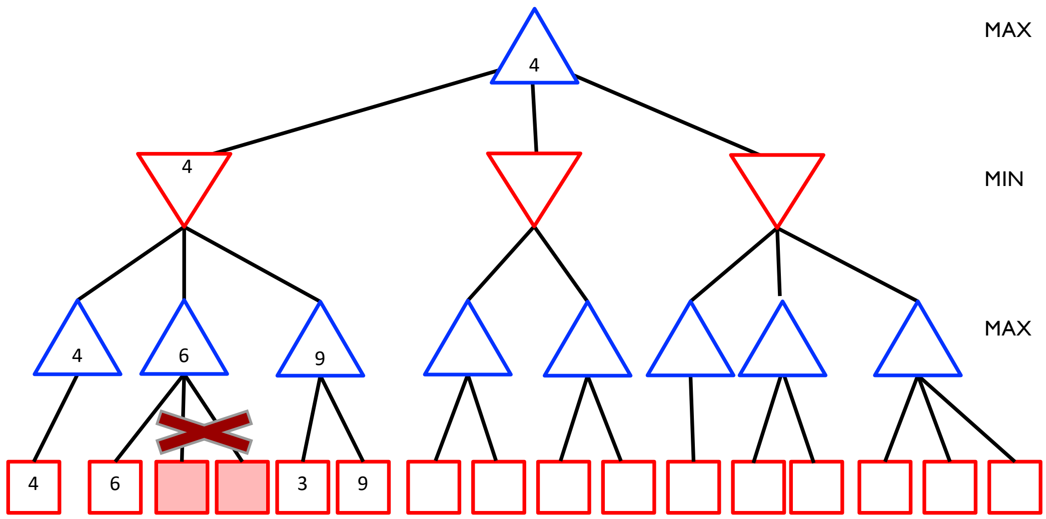
74

Alpha-beta pruning



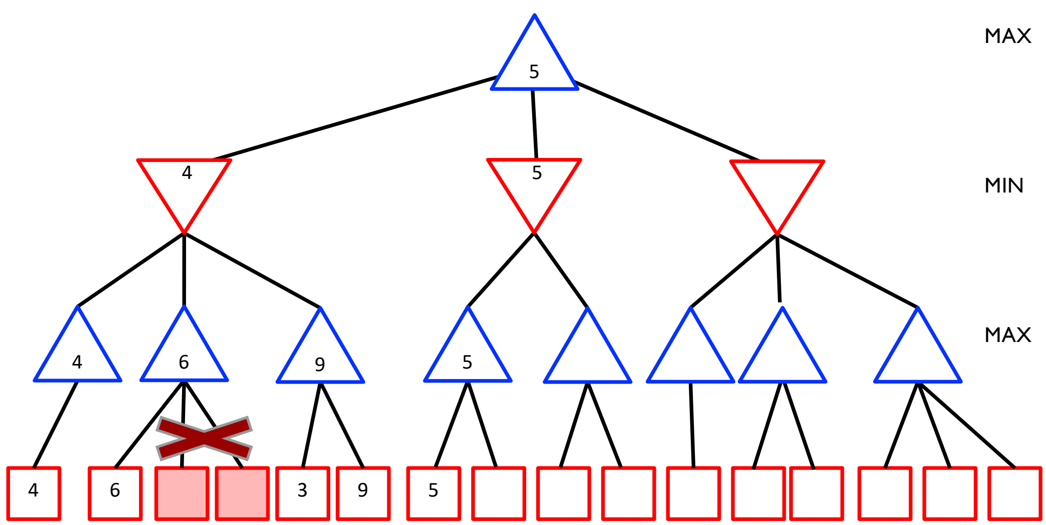
75

Alpha-beta pruning



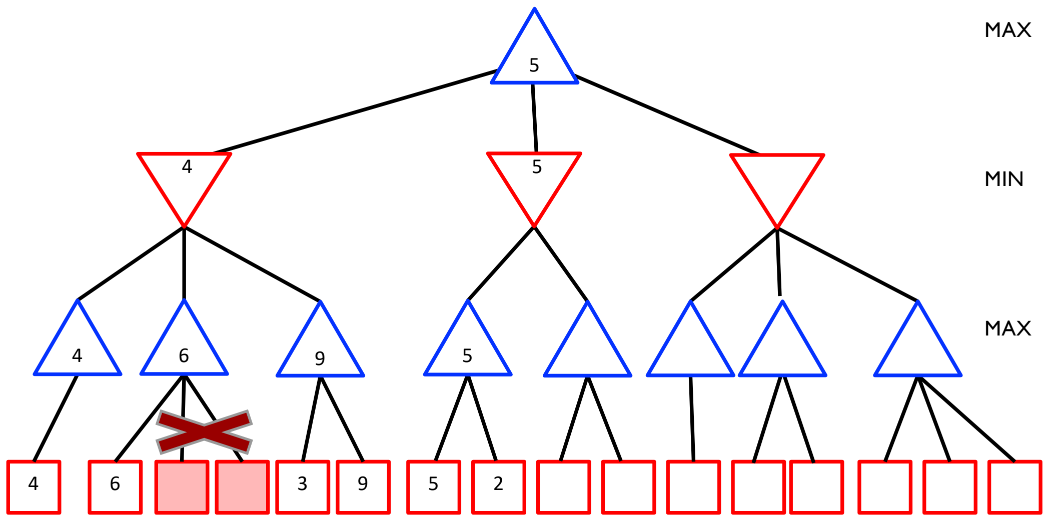
76

Alpha-beta pruning



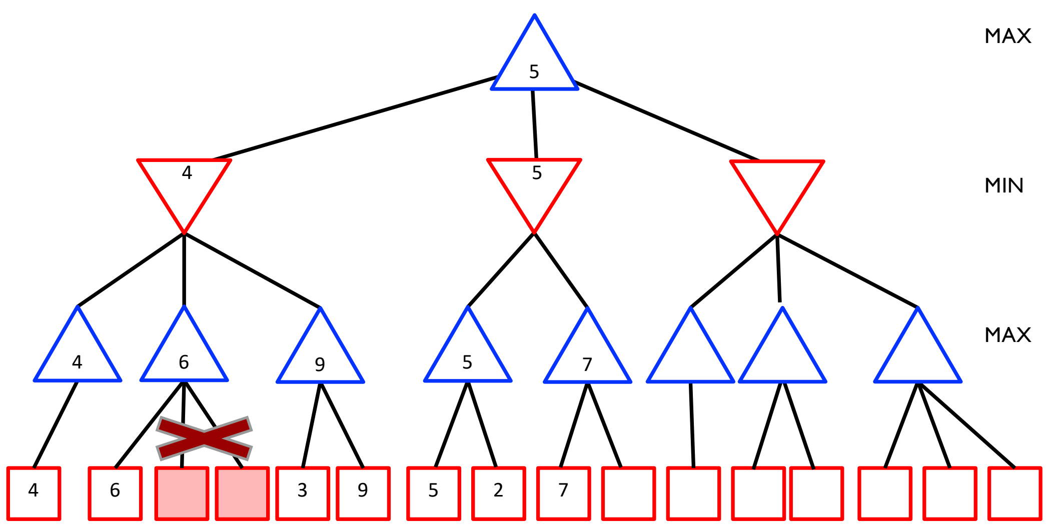
77

Alpha-beta pruning



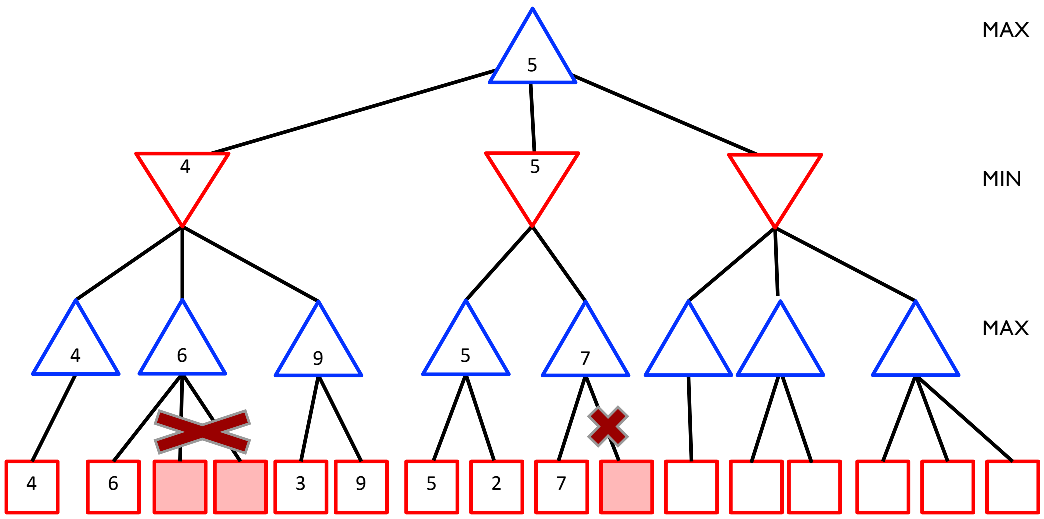
78

Alpha-beta pruning



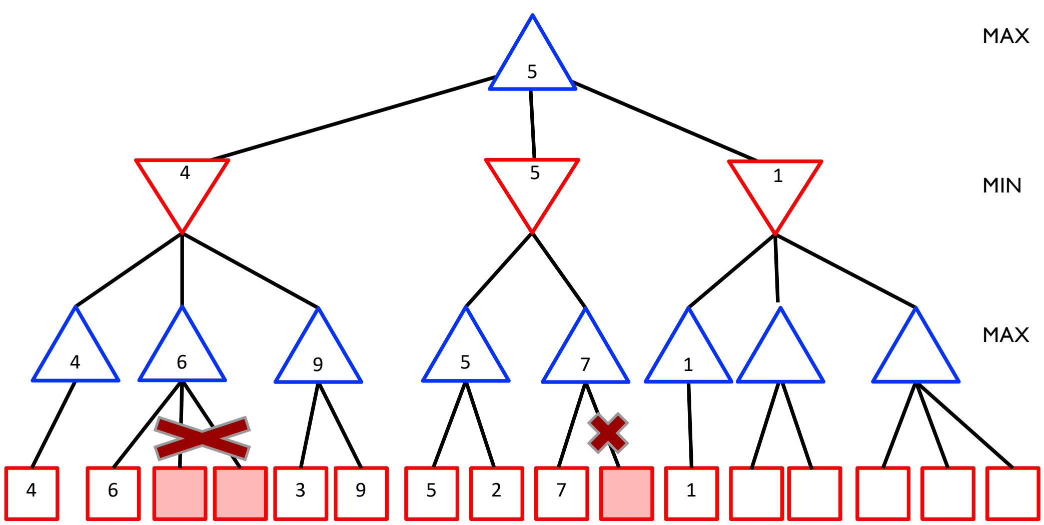
79

Alpha-beta pruning



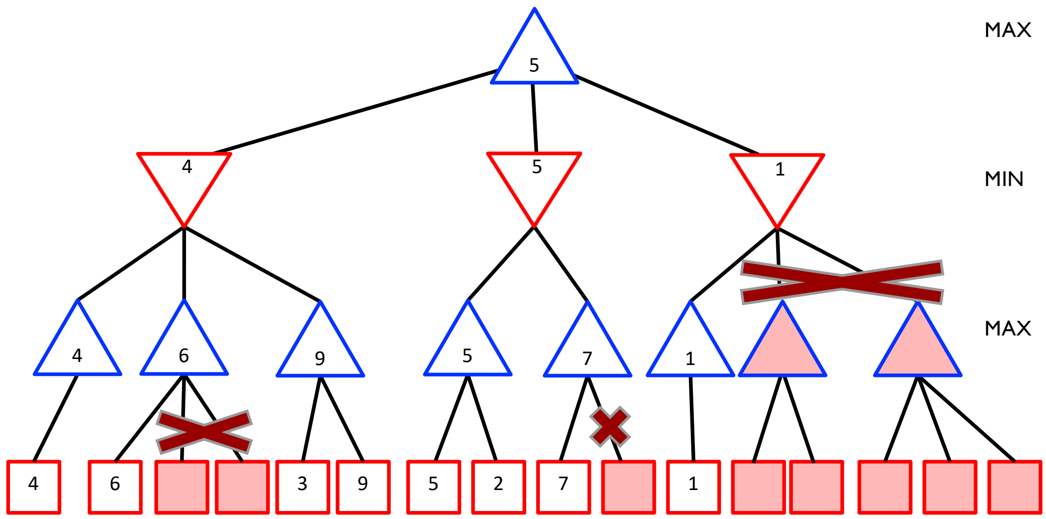
80

Alpha-beta pruning



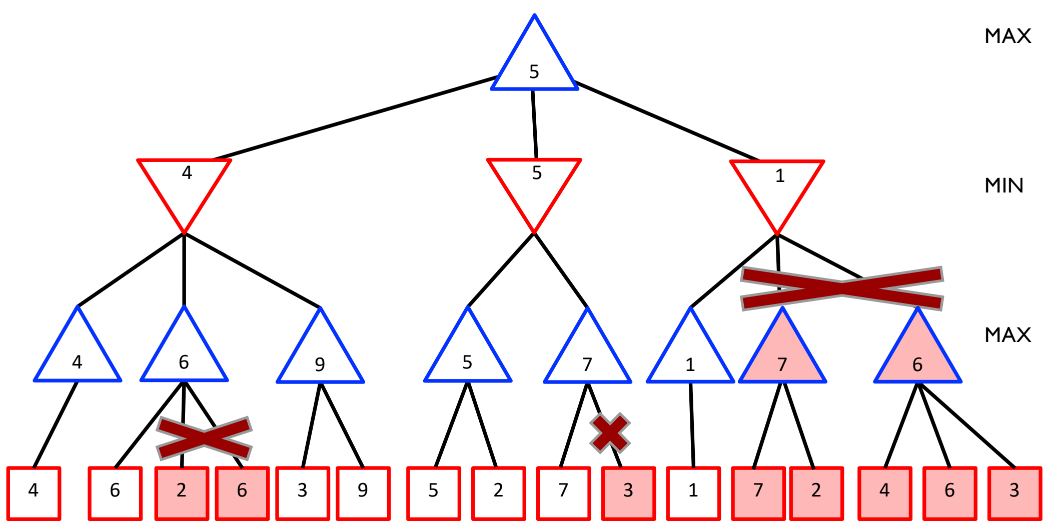
81

Alpha-beta pruning



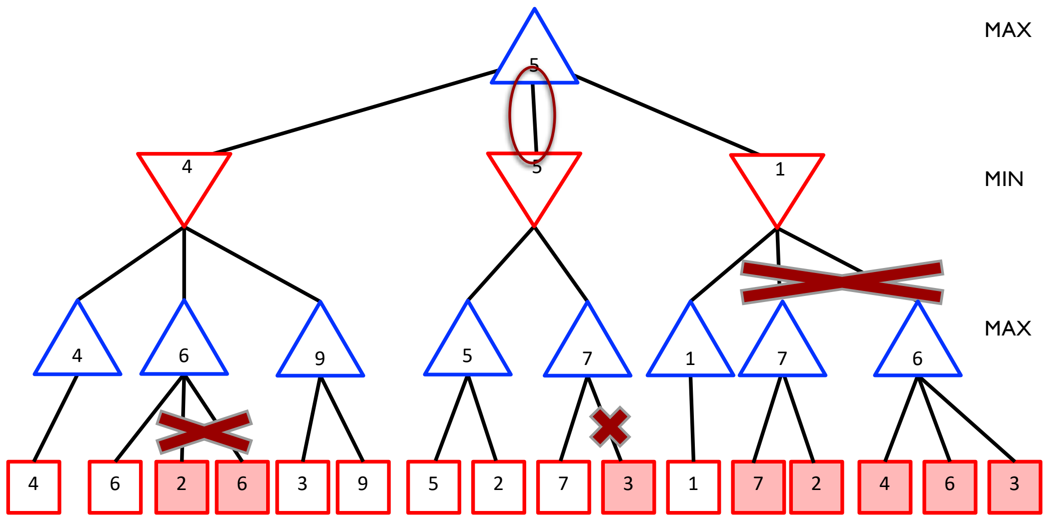
82

Alpha-beta pruning



83

Alpha-beta pruning



84

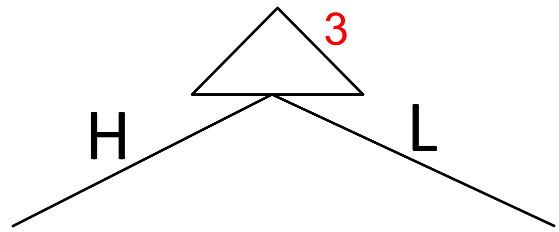
Expectiminimax trees

- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value

85

Expectiminimax trees

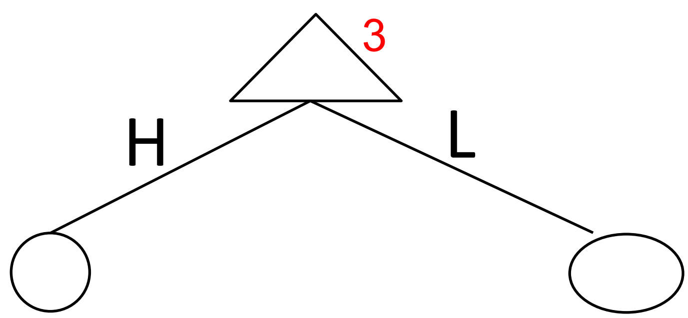
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



86

Expectiminimax trees

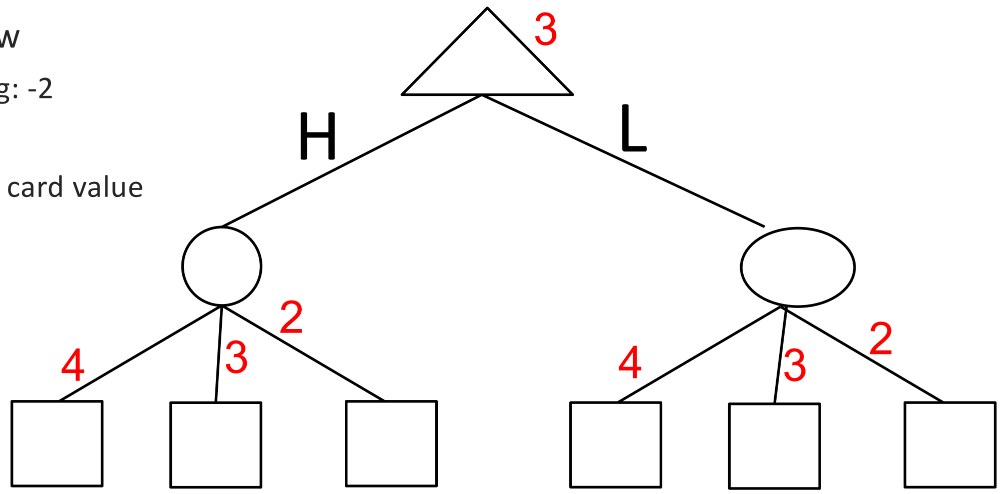
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



87

Expectiminimax trees

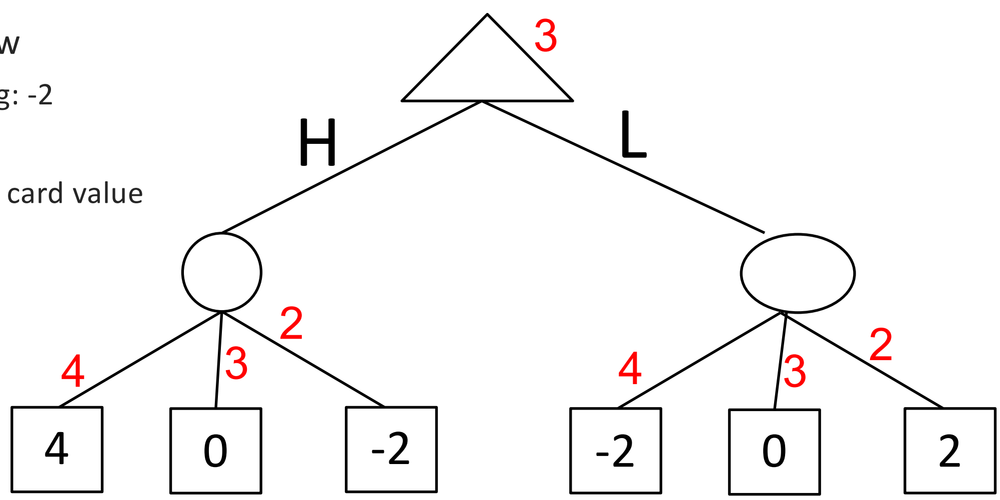
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



88

Expectiminimax trees

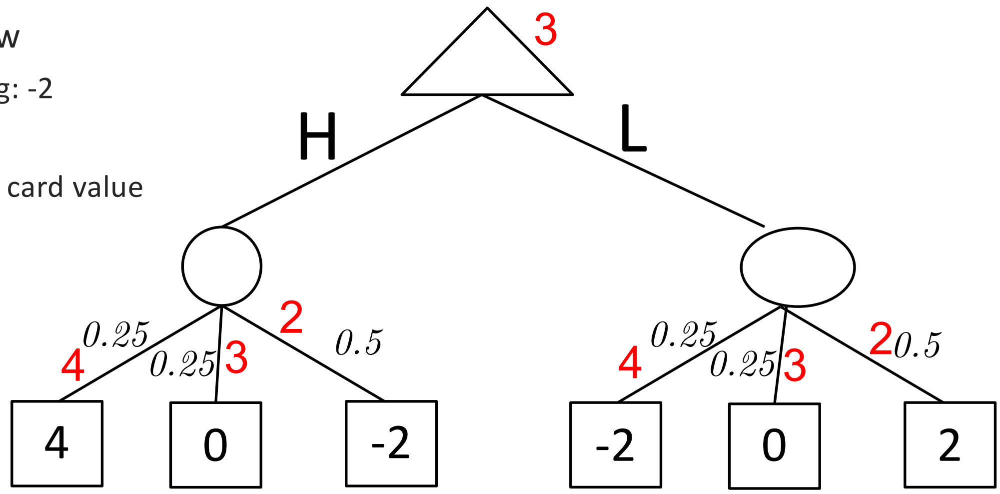
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



89

Expectiminimax trees

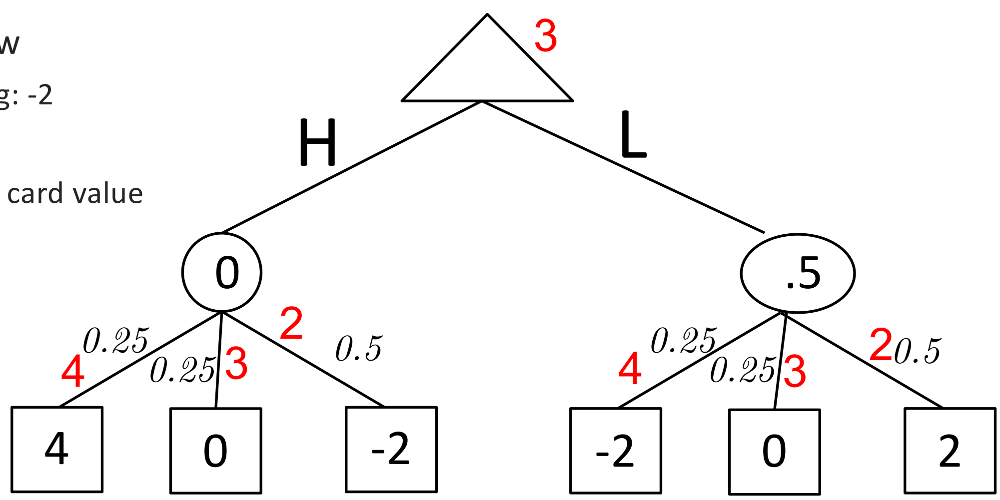
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



90

Expectiminimax trees

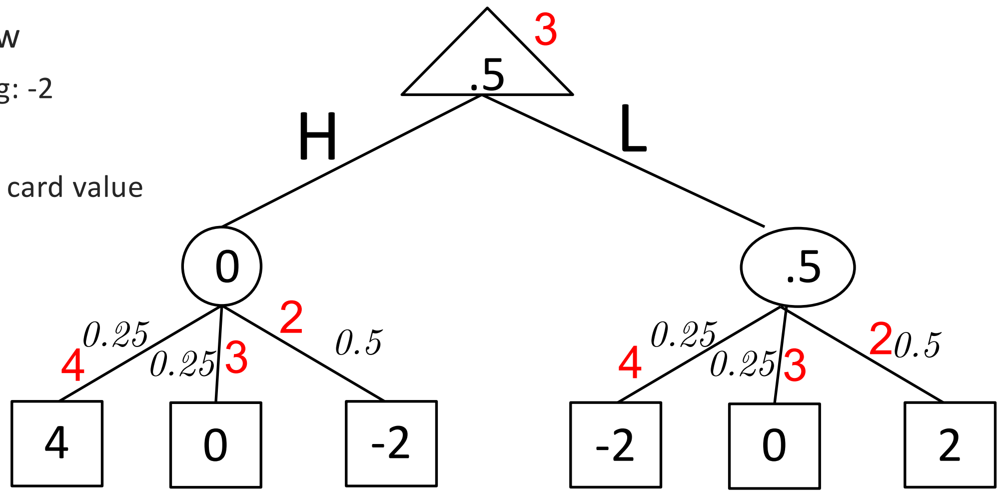
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



91

Expectiminimax trees

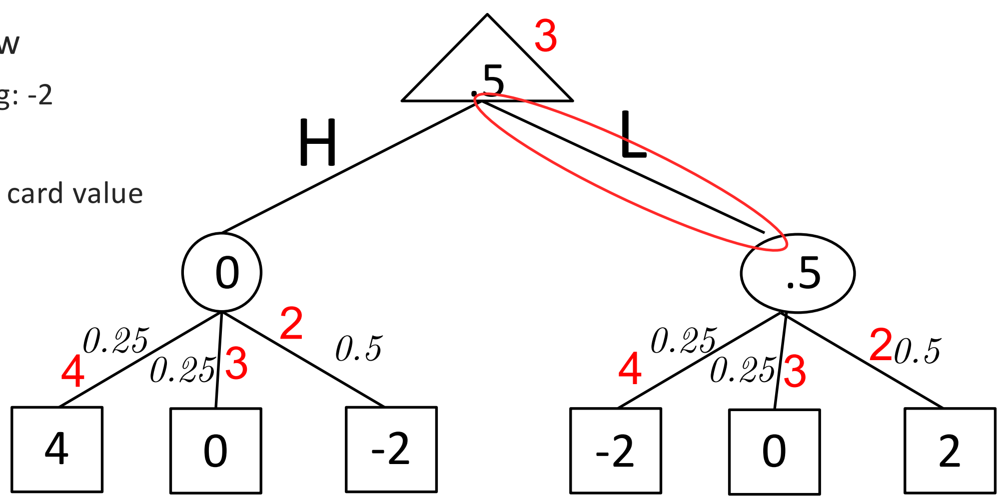
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



92

Expectiminimax trees

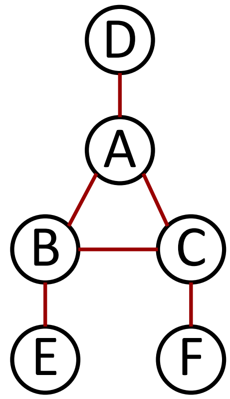
- Cards: 50% 2s, 25% 3s, 25% 4s
- High/Low
  - Wrong: -2
  - Tie: 0
  - Right: card value



93

Constraint Satisfaction

- Variables: A, B, C, D, E, F (each person)
- Domain: 1-6 (seat occupied)
- Constraints: E = B±1, C = B±1, A = D±1, F ≠ C±2, A ≠ B±1, A ≠ C±1, F ≠ C±1
- (I only asked about pairs)
- Forward checking checks *one step* forward: from A to B, C, D
- Legal instantiation: an assignment of values to variables – CBEFDA



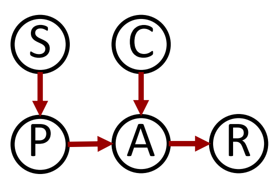
picture: anniejenningspr.com/authorexpertwire/specialty/how-to-get-a-front-row-seat

94

Bayes Belief Net

- Edges indicate *causal* (or *influential*) relationships
- Belief nets are directed
  - Arrows in the graph, not lines
  - Indicate *direction of influence*
- Need to explain what edges denote in your graph
- Idea of *gated influence*
  - That is, cats don't cause runny noses except *through* allergies

Season	**S**
Having a Runny Nose	R
Owning a Cat	C
Pollen Levels High	P
Having Allergies	A



95

Game theory

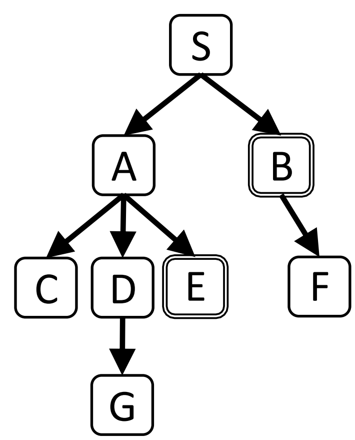
- Zero-sum game: a game with a fixed set of resources/shared outcomes
  - "If I win you lose"
- Pareto optimality: An outcome is Pareto optimal if there is **no other outcome that all players would prefer**
  - A state from which it is impossible to [change] so as to make any one individual better off without making at least one individual worse off
  - s' (Exists.x Ux(s') > Ux(s) → Exists.y Uy(s') < Uy(s))
- Nash equilibrium: Each player's strategy is optimal, **given strategies of the other players**
  - No player benefits by unilaterally changing strategy while others stay fixed

96

Search and iterative deepening

- Search always halts when a goal state is found
- Iterative deepening
  - Depth-first search down to some depth *d*
- Key: redo work as *d* increases

Depth	Current	Frontier
D=1	S	{}
D=2	S	{A B}
	A	{B}
	B	{}



97

Admissible heuristics

- A heuristic in search, h(n), tells you how good a state is
  - A state is “good” if it is closer to achieving a goal
- Takes a state (like current location in map), returns a value
  - Must be applicable to any state

h(

2	8	1
	4	3
7	6	5

) = 8 – that’s bad

h(

1	2	3
8		4
7	6	5

) = 5 – that’s better

98

Admissible heuristics

- A heuristic in search, h(n), tells you how good a state is
  - A state is “good” if it is closer to achieving a goal
- Takes a state (like current location in map), returns a value
  - Must be applicable to any state

h(

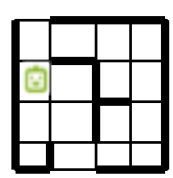
2	8	1
	4	3
7	6	5

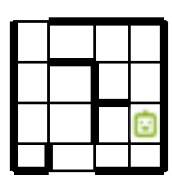
) = 8 – that’s bad

h(

1	2	3
8		4
7	6	5

) = 5 – that’s better

h() = ?

h() = ?

99

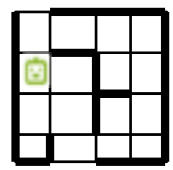
Admissible heuristics

- A heuristic in search, h(n), tells you how good a state is
  - A state is “good” if it is closer to achieving a goal
- Takes a state (like current location in map), returns a value
  - Must be applicable to any state

h(

2	8	1
	4	3
7	6	5

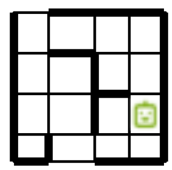
) = 8 – that’s bad

h() = 1

h(

1	2	3
8		4
7	6	5

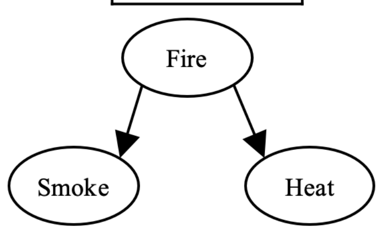
) = 5 – that’s better

h() = 5

100

Belief net calculations

P(Fire)
0.1



Fire	P(Smoke)
T	0.9
F	0.001

Fire	P(Heat)
T	0.99
F	0.0001

- $P(S) = \Sigma_F \Sigma_H P(S \wedge F \wedge H)$
- $P(F|S) = P(F \wedge S) / P(S)$

101

Belief net calculations

- $P(S) = \Sigma_F \Sigma_H P(S \wedge F \wedge H)$
  $= \Sigma_F \Sigma_H P(S \wedge H \mid F) * P(F)$
  $= \Sigma_F \Sigma_H P(S \mid F) * P(H \mid F) * P(F)$
  $= P(S|F) \times P(H|F) \times P(F) +$
  $P(S|F) \times P(\neg H|F) \times P(F) +$
  $P(S|\neg F) \times P(H|\neg F) \times P(\neg F) +$
  $P(S|\neg F) \times P(\neg H|\neg F) \times P(\neg F)$
  $= (.9 \times .99 \times .1) +$
  $(.9 \times .01 \times .1) +$
  $(.001 \times .0001 \times .9) +$
  $(.001 \times .9999 \times .9)$
  $= 0.0909$
- $P(F|S) = P(F \wedge S) / P(S)$
  $= 0.1 \wedge 0.9 / 0.0909$
  $= 0.09 / 0.0909$
  $= 0.99$

102

Reminders and Next Time

- Midterm
  - Rough curve: 60+ = A, 50+ = B, 40+ = C
  - Reminder: 24 hours from handout before we discuss grades
  - I encourage you to go back to materials and seek answers, *before* discussion
- HW3
  - Posted: Filtering example and spreadsheet with worked math
  - Posted: Detailed writeup on information gain
- Questions?

103