Supervised Learning Methods

- k-nearest-neighbor (k-NN)
- Neural networks (ANN)
- Support vector machines (SVM)
- Decision trees

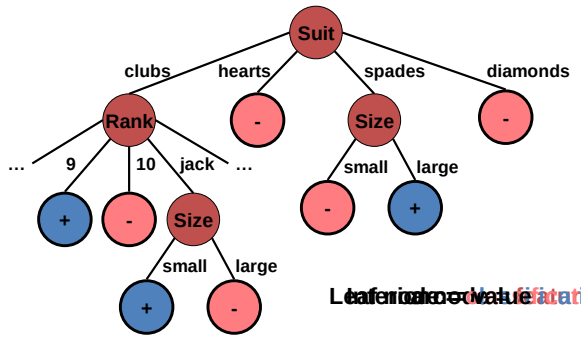
Inductive Concept Learning by Learning Decision Trees

- Goal:
  Build a decision tree for classifying examples as positive or negative instances of a concept
  - is a form of *supervised learning*
  - uses *batch processing* of training examples
  - uses a *preference bias*
    - Learning can be viewed as searching the Hypothesis Space H of possible h functions, $y = h(x)$
    - Preference bias: define a metric for comparing h's so as to determine whether one is better than another

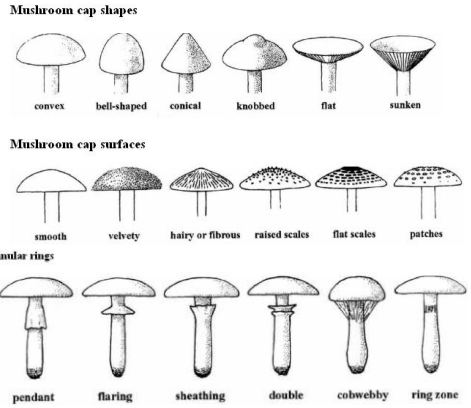
Inductive Concept Learning by Learning Decision Trees

- A **decision tree** is a tree in which:
  - each ***non-leaf node*** has associated with it an attribute/feature
  - each ***leaf node*** has associated with it a classification (class label, e.g., + or -)
  - each ***arc*** has associated with it one of the possible values of the attribute of its parent node (i.e., node from where the arc is directed)

Inductive Concept Learning by Learning Decision Trees



Example: Mushroom Classification



Edible or Poisonous?

http://www.usask.ca/biology/fungi/

Mushroom Features

1. **cap-shape**: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. **cap-surface**: fibrous=f, grooves=g, scaly=y, smooth=s
3. **cap-color**: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. **bruises?**: bruises=t, no=f
5. **odor**: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. **gill-attachment**: attached=a, descending=d, free=f, notched=n
7. …

Classes: edible=e, poisonous=p

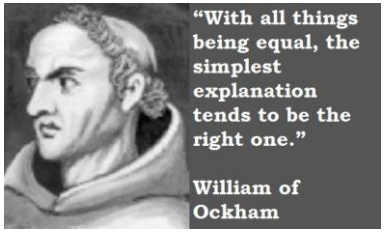
Using a Decision Tree

- A Decision Tree is used as a classifier by taking a given input example, which is given by its feature vector, and:
  1. The attribute at the root node of the tree is interpreted as a question, and the answer is determined by the value of that feature in the input example
  2. Answer determines which child node to move to
  3. Repeat until a leaf node is reached; class label at leaf is the classification given to the input example

Inductive Concept Learning by Learning Decision Trees

- What is the best decision tree?
- Preference Bias: **Ockham's Razor**
  - The ***simplest hypothesis*** that is consistent with all observations is most likely
  - The **smallest decision tree** that correctly classifies all of the training examples is best
- Finding the provably smallest decision tree is an NP-Hard problem, so instead construct one that is "pretty small"

Ockham's Razor



"Everything should be made as simple as possible, but not simpler." – Albert Einstein

Ockham chooses a razor

Decision Tree Construction using a Greedy Algorithm

- Aka **ID3** or **C5.0**
- Top-down construction of the decision tree:
  1. Select the "best attribute" to use for the new node at the current level in the tree
  2. For each possible value of the selected attribute:
     a) Partition the examples using the possible values of this attribute, and assign these subsets of the examples to the appropriate child node
     b) Recursively generate each child node until (ideally) all examples for a node are either all + or all -

Decision Tree Algorithm

buildtree(*examples*, *attributes*, *default*)

/* examples: a list of training examples
attributes: a set of candidate questions, e.g., "what's the value of attribute x_i?"
default: default label prediction, e.g., over-all majority vote */

IF empty(*examples*) THEN return(*default*)

IF (*examples* have same label *y*) THEN return(*y*)

IF empty(*attributes*) THEN return(majority vote in *examples*)

q = best_attribute(*examples, attributes*)

Let there be *n* possible values of attribute *q*

- Create and return an internal node with n children
- The i^{th} child is built by calling

  buildtree({*example*|q=i^{th} value}, *attributes*-{q}, *default*)

Decision Tree Algorithm

- How could the "best attribute" be chosen?
  - **Random:** choose any attribute at random
  - **Least-Values:** choose the attribute with the smallest number of possible values
  - **Most-Values:** choose the attribute with the largest number of possible values
  - **Max-Gain:** choose the attribute that has the largest expected information gain

Information Gain

- How is the information gain determined?
  - **goal:** try to select the attribute that will result in the *smallest expected size* of the sub-trees rooted at its children
  - use information theory

Information Theory

- How many yes/no questions would you expect to ask to determine which number I'm thinking of in the range 1 to 100?

  7
- With each yes/no question in the optimal decision tree at most 1/2 of the elements remaining can be eliminated

  $log_2 100 = 6.64$

Information Theory

- Given a set S of size $|S|$, the expected work required to determine a specific element is:

  $log_2 |S|$
- Call this value the information value of being told the element rather than having to work for it (by asking questions)

Entropy

- At the current node, say there are $n = n_1 + \ldots + n_k$ examples
  - n_1 examples have label y_1
  - n_2 examples have label y_2
  - …
  - n_k examples have label y_k
- What's the **impurity/inhomogeneity/disorder** of the examples at this node?
- Turn it into a game: If I put these examples in a bag, and grab one at random, what is the probability the example has label y_i?

Entropy

- Probability estimated from the given samples:
  - with probability $p_1 = n_1/n$ the example has label y_1
  - with probability $p_2 = n_2/n$ the example has label y_2
  - …
  - with probability $p_k = n_k/n$ the example has label y_k
- $p_1 + p_2 + \ldots + p_k = 1$
- The "outcome" of the draw is a random variable *y* with probability $(p_1, p_2, \ldots, p_k)$
- What's the impurity/disorder of the node? ➔

  What's the uncertainty of *y* in a random drawing?

Entropy

$$H(Y) = \sum_{i=1}^{k} -\Pr(Y = y_i) \log_2 \Pr(Y = y_i)$$

$$= \sum_{i=1}^{k} -p_i \log_2 p_i$$

- Interpretation: The number of yes/no questions (bits) needed on average to pin down the value of *y* in a random drawing

Entropy: *H*(*Y*)

- H measures the information content in **bits** associated with a set of examples
- $0 \leq H(Y)$
  where 0 is no information, and 1 is maximum information (for a 2-class *Y*)
- Bit
  - information needed to answer a yes/no question
  - a real value, *not binary bits*

Information Theory

- Given $S = P \cup N$, where P and N are two disjoint sets, how hard is it to determine which element I am thinking of in S?

if x is in P,
then $log_2 p$ questions needed, where $p = |P|$

if x is in N,
then $log_2 n$ questions needed, where $n = |N|$

Information Theory

- So, the expected number of questions that have to be asked is:
  $(Prob(x \in P) * log_2 p) + (Prob(x \in N) * log_2 n)$
- or, equivalently,
  $(p/(p+n))\ log_2 p + (n/(p+n))\ log_2 n$

Information Extremes

- 2 classes: + and -
- Perfect Balance (Maximum Inhomogeneity):

  given $p_+ = p_- = ½$

  $H(Y) = - ½\ log_2 ½ - ½\ log_2 ½$

  $= -½\ (log_2 1 - log_2 2) - ½\ (log_2 1 - log_2 2)$

  $= -½\ (0 - 1) - ½\ (0 - 1)$

  $= ½ + ½$

  $= 1$ ($\Rightarrow$ the entropy is large)

  A histogram of the frequency distribution of values of *Y* would be nearly flat
- "High Entropy" means *Y* is from a nearly ***uniform*** distribution

Information Extremes

- 2 classes: + and -
- Perfect Homogeneity:

  given $p_+ = 1$ and $p_- = 0$

  $H(Y) = -1\ log_2 1 - 0\ log_2 0$

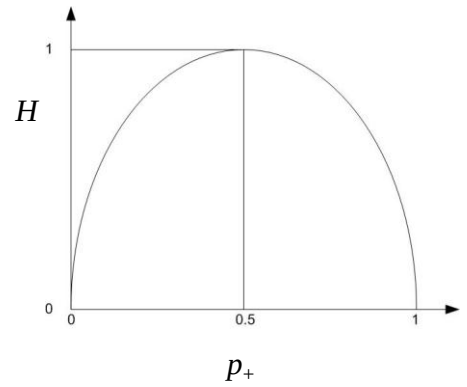
  $= -1\ (0) - ???$

  $= -0 - 0$

  $= 0$ ($\Rightarrow$ no information content)

  A histogram of the frequency distribution of values of *Y* would have many lows and one or two highs
- "Low Entropy" means *Y* is from a varied (peaks and valleys) distribution

Entropy



Entropy

Pr(head) = 0.5
Pr(tail) = 0.5
H = 1

Pr(head) = 0.51
Pr(tail) = 0.49
H = 0.9997

Conditional Entropy

$$H(Y \mid X = v) = \sum_{i=1}^{k} -\Pr(Y = y_i \mid X = v) \log_2 \Pr(Y = y_i \mid X = v)$$

$$H(Y \mid X) = \sum_{v: \text{values of } X} \Pr(X = v) H(Y \mid X = v)$$

- *Y*: a label
- *X*: an attribute (i.e., feature or question)
- *v*: a value of the attribute
- Pr(*Y*|*X*=*v*): conditional probability
- Textbook calls *H*(*Y*|*X*) the ***Remainder***(*X*)

Conditional Entropy: $H(Y \mid X)$

- Weighted sum of the entropy of each subset of the examples partitioned by the possible values of the attribute *X*
- Weighted sum of the entropy at each child node generated by attribute *X*
- *M*easures the total “impurity,” "disorder" or "inhomogeneity“ of the ***children nodes***
- $0 \le H(Y \mid X) \le 1$

Conditional Entropy: $H(Y|X=v)$

Suppose I’m trying to predict output *Y* and I have input *X*

X = College Major

Y = Likes “Gladiator”

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Let’s assume this reflects the true probabilities

From this data we estimate

- *Pr*(*LikeG* = *Yes*) = 0.5
- *Pr*(*Major* = *Math* & *LikeG* = *No*) = 0.25
- *Pr*(*Major* = *Math*) = 0.5
- *Pr*(*LikeG* = *Yes* | *Major* = *History*) = 0

Note:

- *H*(*X*) =
- *H*(*Y*) = 1

Specific Conditional Entropy: $H(Y|X=v)$

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Specific Conditional Entropy:

$H(Y \mid X=v)$ = entropy of Y among only those records in which X has value v

Specific Conditional Entropy: H(Y|X=v)

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Specific Conditional Entropy:

$H(Y \mid X=v)$ = entropy of Y among only those records in which X has value v

Example:

- $H(Y \mid X=Math) = 1$
- $H(Y \mid X=History) = 0$
- $H(Y \mid X=CS) = 0$

Conditional Entropy: H(Y|X)

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Conditional Entropy:

$H(Y \mid X)$ = *average* specific conditional entropy of Y

= if you choose a record at random what will be the conditional entropy of Y, conditioned on that row's value of X

= Expected number of bits to transmit Y if **both sides know the value of X**

$= \Sigma_j Pr(X=v_j)\, H(Y \mid X = v_j)$

Conditional Entropy

X = College Major

Y = Likes "Gladiator"

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

Definition of Conditional Entropy:

$H(Y \mid X)$ = average conditional entropy of Y

$= \Sigma_j Pr(X=v_j)\, H(Y \mid X = v_j)$

Example:

v_j	$Pr(X=v_j)$	$H(Y \mid X=v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

$H(Y|X) = 0.5 * 1 + 0.25 * 0 + 0.25 * 0 = 0.5$

Information Gain

- **Information gain**, or **mutual information**

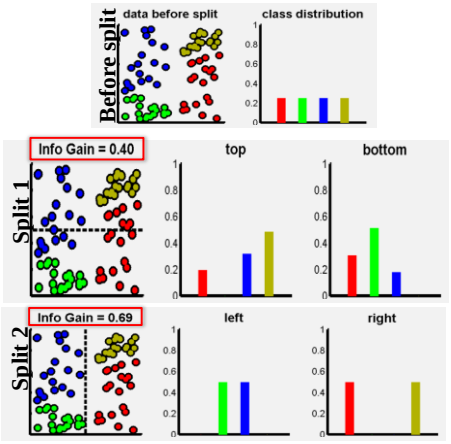
$$I(Y;X) = H(Y) - H(Y \mid X)$$

- Measures the *difference* in entropy of a node and the entropy *remaining* after the node's examples are "split up" between the children using a chosen attribute
- *I(Y; X) = I* must transmit *Y.* How many bits on average would it save me if both ends of the line knew *X*?
- Choose the attribute (i.e., feature or question) *X* that ***maximizes*** *I*(*Y*; *X*)
- Textbook calls *I*(*Y*; *X*) the ***Gain***(*X*)

Using Information Gain to Select the Best Attribute

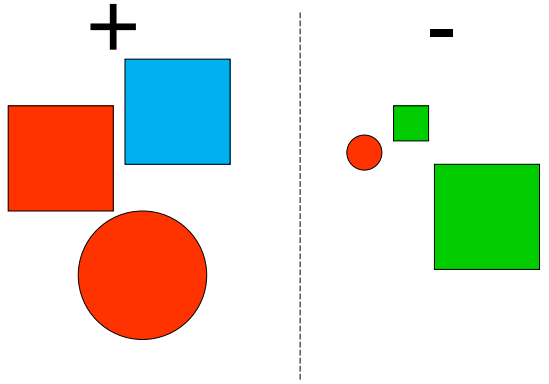
- Goal: Construct a small decision tree that correctly classifies the training examples
- Why would high information gain be desirable?
  - means more of the examples are the same class in the child nodes
  - the decision trees rooted at each child that are needed to differentiate between the classes should be small

Using Information Gain



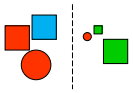
Example

- Features: color, shape, size
- What's the best attribute for the root?



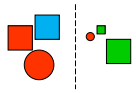
The Training Set

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



H(class)=
H(class | color)=

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



H(class)= *H*(3/6,3/6) = 1
H(class | color)= 3/6 * *H*(2/3,1/3) + 1/6 * *H*(1,0) + 2/6 * *H*(0,1)

3 out of 6 are red

2 of the red are +

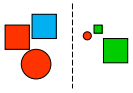
1 out of 6 is blue

blue is +

2 out of 6 are green

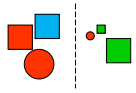
green is -

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



H(class)= *H*(3/6,3/6) = 1
H(class | color)= 3/6 * *H*(2/3,1/3) + 1/6 * *H*(1,0) + 2/6 * *H*(0,1)
I(class; color) = *H*(class) – *H*(class | color) = 0.54 bits

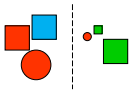
Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



H(class)= *H*(3/6,3/6) = 1
H(class | shape)= 4/6 * *H*(1/2, 1/2) + 2/6 * *H*(1/2,1/2)
I(class; shape) = *H*(class) – *H*(class | shape) = 0 bits

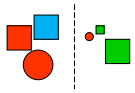
Shape tells us nothing about the class!

Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



H(class)= H(3/6,3/6) = 1
H(class | size)= 4/6 * H(3/4, 1/4) + 2/6 * H(0,1)
I(class; size) = H(class) – H(class | size) = 0.46 bits

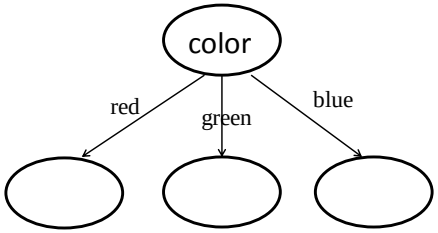
Example	Color	Shape	Size	Class
1	Red	Square	Big	+
2	Blue	Square	Big	+
3	Red	Circle	Big	+
4	Red	Circle	Small	-
5	Green	Square	Small	-
6	Green	Square	Big	-



I(class; color) = H(class) – H(class | color) = 0.54 bits
I(class; shape) = H(class) – H(class | shape) = 0 bits
I(class; size) = H(class) – H(class | size) = 0.46 bits

➔ Select color as the best attribute at the root

What's the Next Step?



Selecting the Best Attribute

- The best attribute for a node is the attribute A (of those candidates available for that node) with:
  - Maximum Information Gain, or
  - Minimum Conditional Entropy
    - Since at a given node, since $H(Y)$ is fixed

Decision Tree Algorithm

buildtree(*examples*, *attributes*, *default*)

/* examples: a list of training examples
attributes: a set of candidate questions, e.g., "what's the value of attribute x_i?"
default: default label prediction, e.g., over-all majority vote */

IF empty(*examples*) THEN return(*default*)

IF (*examples* have same label *y*) THEN return(*y*)

IF empty(*attributes*) THEN return(majority vote in *examples*)

q = best_attribute(*examples, attributes*)

Let there be *n* possible values of attribute *q*

- Create and return an internal node with n children
- The i^{th} child is built by calling

  buildtree({*example*|q=i^{th} value}, *attributes*-{q}, *default*)

Case Studies

- Decision trees have been shown to be at least as accurate as human experts
- 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
- Cessna designed a flight controller using 90,000 examples and 20 attributes per example

Expressiveness of Decision Trees

- Assume all inputs are Boolean and all outputs are Boolean
- What is the class of Boolean functions that are possible to represent by decision trees?
- Answer: All Boolean functions!

Simple proof:

1. Take any Boolean function
2. Convert it into a truth table
3. Construct a decision tree in which each row of the truth table corresponds to one path through the decision tree

Overfitting a Decision Tree

- In general, **overfitting means finding "meaningless" regularity in data**
- Noisy Data: "noise" could be in the examples:
  - examples have the same attribute values, but different classifications
  - classification is wrong
  - attributes values are incorrect because of errors getting or preprocessing the data
  - irrelevant attributes

Overfitting a Decision Tree

Five inputs, all bits, are generated in all 32 possible combinations

Output ***y*** = copy of e, except a random 25% of the records have ***y*** set to the opposite of e

32 records

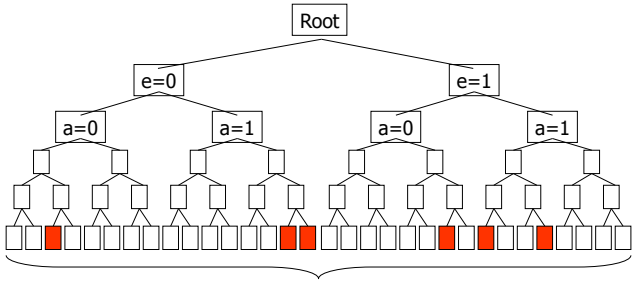
a	*b*	*c*	*d*	*e*	*y*
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

Overfitting a Decision Tree

- The test set is constructed similarly
  - ***y*=*e***, but 25% the time we corrupt it by ***y*** = **1-*e***
  - The corruptions in training and test sets are *independent*
- The training and test sets are the same, except
  - Some ***y***'s are corrupted in training, but not in test
  - Some ***y***'s are corrupted in test, but not in training

Overfitting a Decision Tree

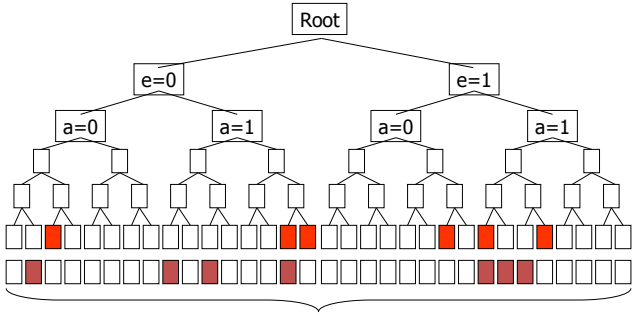
- Suppose we build a full tree on the **training set**



Training set accuracy = 100% (all leaf nodes contain exactly 1 example)
25% of these training leaf node labels will be corrupted (≠***e***)

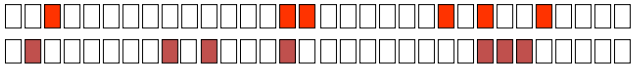
Overfiting a Decision Tree

- Next, classify the **test data** with the tree



25% of the test examples are corrupted – independent of training data

Overfitting a Decision Tree



On average:

- ¾ training data uncorrupted
  - ¾ of these are uncorrupted in test – correct labels
  - ¼ of these are corrupted in test – wrong
- ¼ training data corrupted
  - ¾ of these are uncorrupted in test – wrong
  - ¼ of these are also corrupted in test – correct labels
- Test accuracy = (¾ * ¾) + (¼ * ¼) = 5/8 = 62.5%

Overfitting a Decision Tree

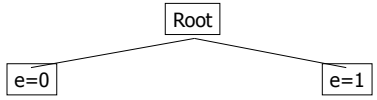
- But if we knew ***a***,***b***,***c***,***d*** are **irrelevant attributes** and don't use them in the tree…

Pretend they don't exist

a	*b*	*c*	*d*	*e*	*y*
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	1
:	:	:	:	:	:
1	1	1	1	1	1

Overfitting a Decision Tree

- The tree would be:



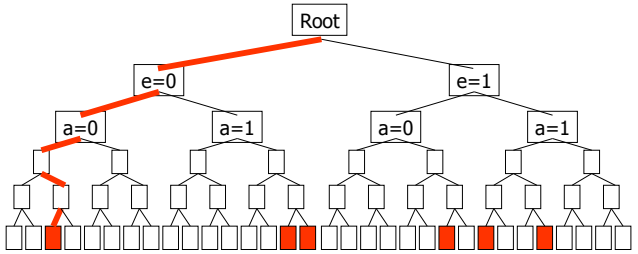
In training data, about ¾ ***y***'s are 0 here. Majority vote predicts ***y***=0

In training data, about ¾ ***y***'s are 1 here. Majority vote predicts ***y***=1

In test data, ¼ ***y***'s are different from ***e*** because they were corrupted, and ¾ ***y***'s will be correct, so test accuracy = 75%, which is better than when using more (meaningless) attributes (= 62.5%)

Overfitting a Decision Tree

- In the full tree, we ***overfit*** by learning non-existent relations (noise)



Extensions to Decision Tree Learning Algorithm

- **Overfitting**
  - meaningless regularity is found in the data
  - irrelevant attributes confound the true, important, distinguishing features
  - fix by pruning some nodes in the decision tree

Avoiding Overfitting: **Pruning**

Pruning with a **tuning set**

1. Randomly split the training data into TRAIN and TUNE, say 70% and 30%
2. Build a full tree using only TRAIN set
3. Prune the tree using the TUNE set

Pruning using a Greedy Algorithm

Prune(tree T, TUNE set)

1. Compute T's accuracy on TUNE, call it A(T)
2. For every internal node N in T:
   a) New tree T_N = copy of T, but prune (delete) the subtree under N
   b) N becomes a leaf node in T_N. The label is the majority vote of TRAIN examples reaching N
   c) $A(T_N)$ = T_N's accuracy on TUNE
3. Let T* be the tree (among the T_N's and T) with the largest A() Set T = T^* /* prune */
4. Repeat from Step 1 until no more improvement
5. Return T

Extensions to Decision Tree Learning: Real-valued Features

- What if some (or all) of the features, x_1, x_2, …, x_k, are real-valued?
- Example: x_1=height (in inches)
- Idea 1: Branch on each possible numerical value

Extensions to Decision Tree Learning: Real-valued Features

- What if some (or all) of the features, x_1, x_2, …, x_k, are real-valued?
- Example: x_1=height (in inches)
- Idea 1: Branch on each possible numerical value
  - fragments the training data and prone to overfitting
- Idea 2: Use questions of the form of ($x_1 > t$?), where t is a threshold

Extensions to Decision Tree Learning: Missing Data

- learning: replace with most likely value
- learning: use *NotKnown* as a value
- classifying: follow arc for all values and weight each by the frequency of examples following that arc

Extensions to Decision Tree Learning Algorithm

- **Generation of rules**

  for each path from the root to a leaf
  - the rule's antecedent is the attribute tests
  - the consequent is the classification at leaf node

```
if (Size = small && Suit = hearts) class = '+';
```

  - Constructing these rules yields an interpretation of the tree's meaning

Decision Trees Summary

- One of the most widely used learning methods in practice
- Can out-perform human experts in many problems

Decision Trees Summary

- Strengths
  - fast
  - simple to implement
  - well founded in information theory
  - can convert result to a set of easily interpretable rules
  - empirically valid in many commercial products
  - handles noisy data
  - scales well

Decision Trees Summary

- Weaknesses
  - Univariate splits/partitions using only one attribute at a time, which limits types of possible trees
  - large decision trees may be hard to understand
  - requires fixed-length feature vectors
  - non-incremental (i.e., batch method)

Combining Classifiers: Ensemble Methods

- **Aggregation of predictions of *multiple* classifiers with the goal of improving accuracy by reducing the variance of an estimated prediction function**
- Mixture of experts
- Combining multiple classifiers often produces higher accuracy than any individual classifier

Example: *Netflix Prize Competition*

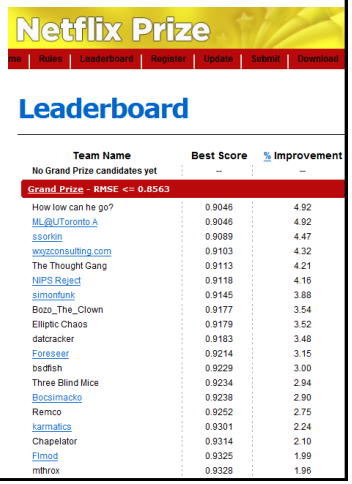
Began October 2006

- Supervised learning task
  - Training data is a set of users and ratings (1,2,3,4,5 stars) those users have given to movies
  - Construct a classifier that given a user and an unrated movie, correctly classifies that movie as either 1, 2, 3, 4, or 5 stars
- $1 million prize for a 10% improvement over Netflix's current movie recommender/classifier (MSE = 0.9514)

Slide by T. Holloway

Just 3 weeks after it began, at least 40 teams could beat the Netflix classifier

Top teams showed about 5% improvement

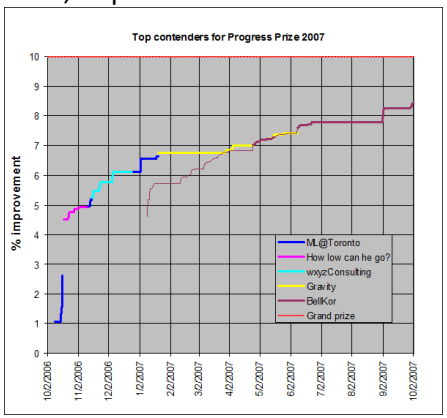


Leaderboard

	Team Name	Best Score	% Improvement
	No Grand Prize candidates yet	--	--
	Grand Prize - RMSE <= 0.8563		
	How low can he go?	0.9046	4.92
	ML@UToronto A	0.9046	4.92
	ssorkin	0.9089	4.47
	wxyzconsulting.com	0.9103	4.32
	The Thought Gang	0.9113	4.21
	NIPS Reject	0.9118	4.16
	simonfunk	0.9145	3.88
	Bozo_The_Clown	0.9177	3.54
	Elliptic Chaos	0.9179	3.52
	datcracker	0.9183	3.48
	Foreseer	0.9214	3.15
	bsdfish	0.9229	3.00
	Three Blind Mice	0.9234	2.94
	Bocsimacko	0.9238	2.90
	Remco	0.9252	2.75
	karmatics	0.9301	2.24
	Chapelator	0.9314	2.10
	Flmod	0.9325	1.99
	mthrox	0.9328	1.96

Slide by T. Holloway

However, improvement slowed…



from http://www.research.att.com/~volinsky/netflix/

Slide by T. Holloway

Ensemble methods to the rescue…

Rookies

"Thanks to Paul Harrison's collaboration, a simple mix of our solutions improved our result from 6.31 to 6.75"

--	No Progress Prize candidates yet	--	--
	Progress Prize - RMSE <= 0.8625		
1	BellKor	0.8705	8.50
	Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell		
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	[Rookies]	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

Arek Paterek

"My approach is to **combine the results of many methods** (also two-way interactions between them) using linear regression on the test set. The best method in my ensemble is regularized SVD with biases, post processed with kernel ridge regression"

http://rainbow.mimuw.edu.pl/~ap/ap_kdd.pdf

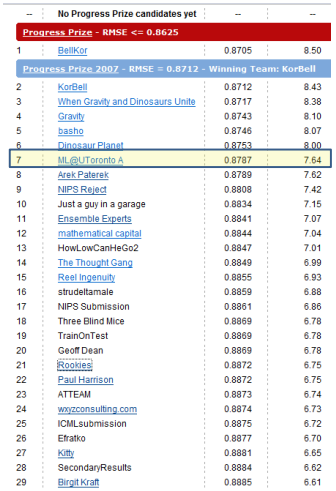
--	No Progress Prize candidates yet	--	--
	Progress Prize - RMSE <= 0.8625		
1	BellKor	0.8705	8.50
	Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell		
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	[Rookies]	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

U of Toronto

"When the predictions of **multiple** RBM models and **multiple** SVD models are linearly combined, we achieve an error rate that is well over 6% better than the score of Netflix's own system."

http://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf



--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

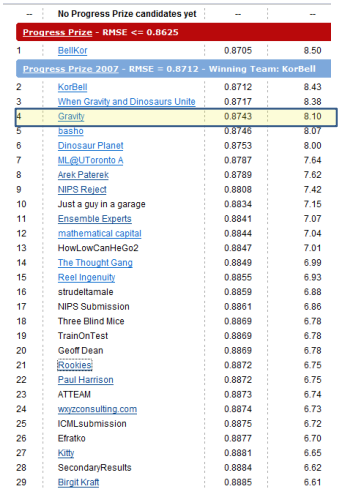
Slide by T. Holloway

Gravity

Table 5: Best results of single approaches and their combinations

Method/Combination	RMSE
MF	0.9190
NB	0.9313
CL	0.9606
NB + CL	0.9275
MF + CL	0.9137
MF + NB	0.9089
MF + NB + CL	0.9089

home.mit.bme.hu/~gtakacs/download/**gravity**.pdf

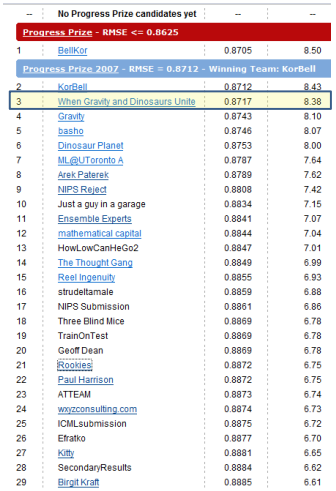


--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

When Gravity and Dinosaurs Unite

"Our common team blends the result of team Gravity and team Dinosaur Planet."



--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

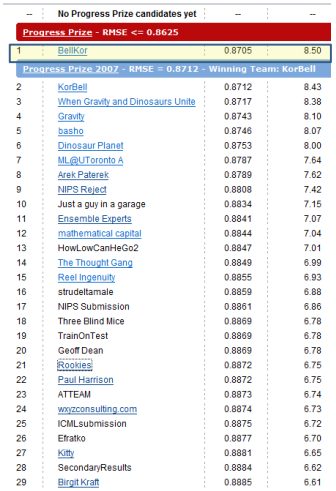
Slide by T. Holloway

BellKor / KorBell

The winning team was from AT&T:

"Our final solution (RMSE=0.8712) consists of blending 107 individual results. "

An 8.5% improvement over Netflix



--	No Progress Prize candidates yet	--	--
Progress Prize - RMSE <= 0.8625			
1	BellKor	0.8705	8.50
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
2	KorBell	0.8712	8.43
3	When Gravity and Dinosaurs Unite	0.8717	8.38
4	Gravity	0.8743	8.10
5	basho	0.8746	8.07
6	Dinosaur Planet	0.8753	8.00
7	ML@UToronto A	0.8787	7.64
8	Arek Paterek	0.8789	7.62
9	NIPS Reject	0.8808	7.42
10	Just a guy in a garage	0.8834	7.15
11	Ensemble Experts	0.8841	7.07
12	mathematical capital	0.8844	7.04
13	HowLowCanHeGo2	0.8847	7.01
14	The Thought Gang	0.8849	6.99
15	Reel Ingenuity	0.8855	6.93
16	strudeltamale	0.8859	6.88
17	NIPS Submission	0.8861	6.86
18	Three Blind Mice	0.8869	6.78
19	TrainOnTest	0.8869	6.78
20	Geoff Dean	0.8869	6.78
21	Rookies	0.8872	6.75
22	Paul Harrison	0.8872	6.75
23	ATTEAM	0.8873	6.74
24	wxyzconsulting.com	0.8874	6.73
25	ICMLsubmission	0.8875	6.72
26	Efratko	0.8877	6.70
27	Kitty	0.8881	6.65
28	SecondaryResults	0.8884	6.62
29	Birgit Kraft	0.8885	6.61

Slide by T. Holloway

Why Combine Classifiers?

- **Statistical**: When training data is small relative to size of hypothesis/classifier space, there are many possible classifiers that fit the data; "averaging" their results reduces risk of picking wrong classifier
- **Computational**: Small training set + local search means hard to find "true" classifier; ensemble simulates multiple random restarts to obtain multiple classifiers
- **Representational**: True classifier may not be representable in the hypothesis space of a method, but some (weighted) combination of hypotheses expands the space of representable functions

How to Combine Classifiers?

Given a test example, classify it using each classifier and report as the output class the **majority** (for a 2-class problem) or **mode** classification

Intuition

Majority Vote Classifier

Suppose we have 5 completely independent classifiers

- If accuracy is 70% for each, combined accuracy is: $10(.7^3)(.3^2) + 5(.7^4)(.3) + (.7^5)$
  - **83.7% majority vote accuracy**
- 101 such classifiers
  - **99.9% majority vote accuracy**

Slide by T. Holloway

When is an Ensemble Better?

- Necessary and sufficient conditions for an ensemble to be *more accurate* than individual classifiers, is when each individual classifier is:
  - **Accurate**: error rate is better than random guessing
  - **Diverse**: Classifiers make errors on different examples, i.e., they are at least somewhat uncorrelated

Ensemble Strategies

Boosting

- Sequential production of classifiers, where each classifier is dependent on the previous one
- Make examples misclassified by current classifier *more* important in the next classifier

Bagging

- Create classifiers using different training sets, where each training set is created by "bootstrapping," i.e., drawing examples (with replacement) from all possible training examples

Slide by T. Holloway

Bagging

- Given *N* training examples, generate separate training sets by choosing *n* examples with replacement from all *N* examples
- Called "**taking a bootstrap sample**" or "randomizing the training set"
- Construct a classifier using the *n* examples in current training set
- Calculate error using rest of training examples
- Repeat for multiple classifiers

Bagging Example (Opitz, 1999)

$N = 8, n = 8$

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

Bagging with Decision Trees

- For each training set, build a separate decision tree
- Take majority/mode vote from all the decision trees to determine the output classification of a given testing example

Random Forests

aka Decision Forest, Classification Forest

2 Main Ideas:

1. **Bagging**: Use random samples of the training examples to construct the classifiers
2. **Randomized Node Optimization**: Each time a node is split, only a randomly chosen subset of the features/attributes are considered

Random Forests

For each tree,

1. Choose a training set by choosing *n* times with replacement from all *N* available training examples
2. At each node of decision tree during construction, choose a random subset of *m* ***features/attributes*** from the total number, *M*, of possible attributes (*m* << *M*)
3. Select best attribute at node using Max-Gain

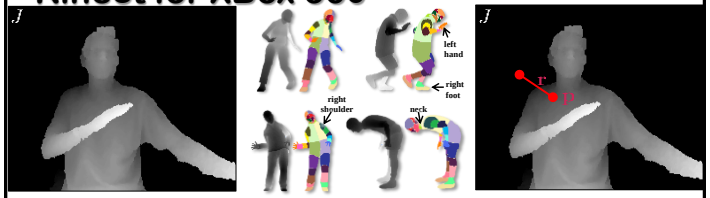
- No tree pruning
- Doesn't overfit

Classification Error

Data set	Adaboost	Selection	Forest-RI single input	One tree
Glass	22.0	20.6	21.2	36.9
Breast cancer	3.2	2.9	2.7	6.3
Diabetes	26.6	24.2	24.3	33.1
Sonar	15.6	15.9	18.0	31.7
Vowel	4.1	3.4	3.3	30.4
Ionosphere	6.4	7.1	7.5	12.7
Vehicle	23.2	25.8	26.4	33.1
German credit	23.5	24.4	26.2	33.3
Image	1.6	2.1	2.7	6.4
Ecoli	14.8	12.8	13.0	24.5
Votes	4.8	4.1	4.6	7.4
Liver	30.7	25.1	24.7	40.6
Letters	3.4	3.5	4.7	19.8
Sat-images	8.8	8.6	10.5	17.2
Zip-code	6.2	6.3	7.8	20.6
Waveform	17.8	17.2	17.3	34.0
Twonorm	4.9	3.9	3.9	24.7
Threenorm	18.8	17.5	17.5	38.4
Ringnorm	6.9	4.9	4.9	25.7

Breiman, Leo (2001). "Random Forests". Machine Learning 45 (1), 5-32

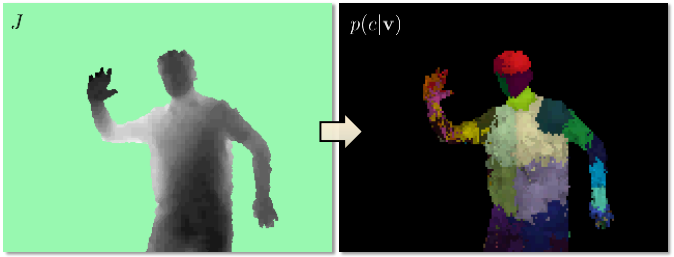
Body Tracking in Microsoft Kinect for XBox 360



Input depth image | Training labelled data | Visual features

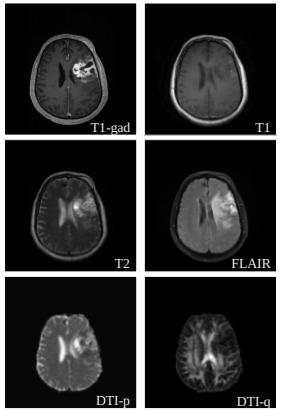
J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, **Real-Time Human Pose Recognition in Parts from a Single Depth Image**, *Proc. Computer Vision and Pattern Recognition Conference*, 2011

Body Tracking in Microsoft Kinect for XBox 360



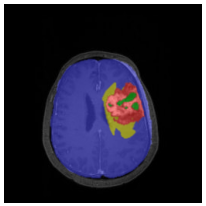
Input depth image (bg removed) | Inferred body parts posterior

Segmentation of Tumors



MRI input data

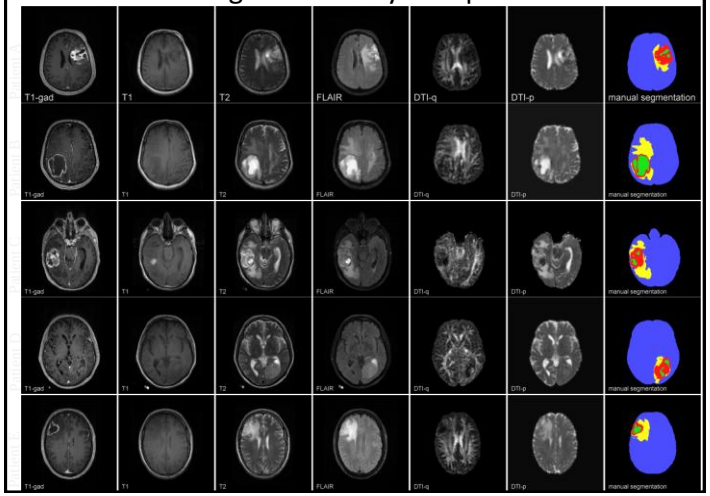
Segmentation of tumorous tissues:



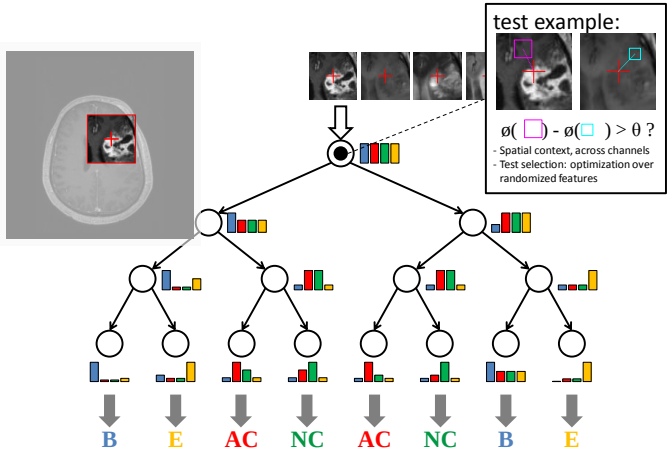
---- Active cells
---- Necrotic core
---- Edema
---- Background

D. Zikic, B. Glocker, E. Konukoglu , A. Criminisi, J. Shotton, C. Demiralp, O. Thomas, T. Das, R. Jena and S. Price, **Decision Forests for Tissue-specific Segmentation of High-grade Gliomas in Multi-Channel MR,** *Proc. MICCAI,* 2012

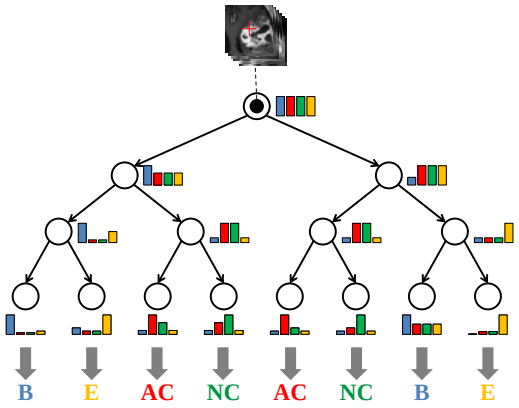
Challenge: Variability of Input Data



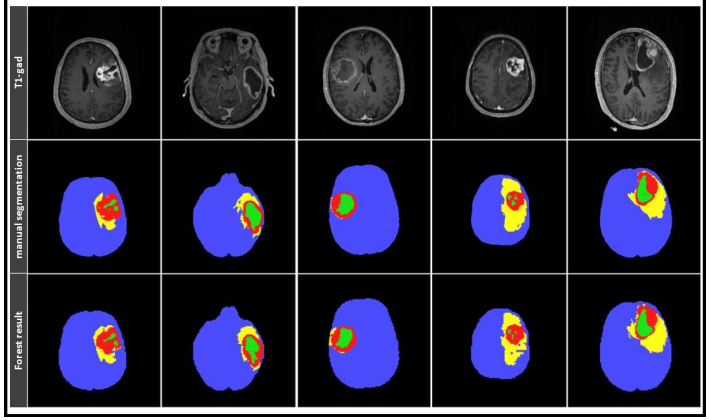
Training the Random Forest



Testing using the Random Forest



Glioblastoma Segmentation



Random Forest Summary

- Advantages
  - One of the most accurate learning algorithms
  - Efficient
  - Can handle thousands of attributes
- Disadvantages
  - Difficult to interpret (compared to decision trees)