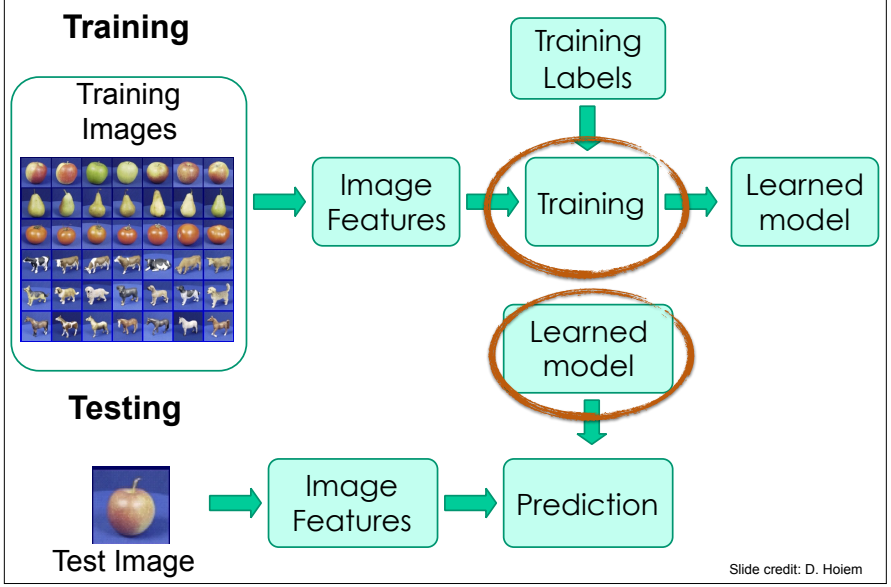
Decision trees

Subhransu Maji

CMPSCI 670: Computer Vision

November 1, 2016

Recall: Steps



Slide credit: D. Hoiem

The decision tree model of learning

- Classic and natural model of learning
- Question: Will an unknown student enjoy an unknown course?
  - **You:** Is the course under consideration in Systems?
  - **Me:** Yes
  - **You:** Has this student taken any other Systems courses?
  - **Me:** Yes
  - **You:** Has this student liked most previous Systems courses?
  - **Me:** No
  - **You:** *I predict this student will not like this course.*
- Goal of learner: Figure out what questions to ask, and in what order, and what to predict when you have answered enough questions

Learning a decision tree

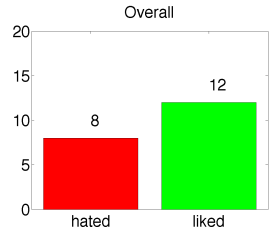
- Recall that one of the ingredients of learning is training data
  - I'll give you *(x, y)* pairs, i.e., set of (attributes, label) pairs
  - We will simplify the problem by
    - {0,+1, +2} as "liked"
    - {-1,-2} as "hated"
- Here:
  - Questions are features
  - Responses are feature values
  - Rating is the label
- Lots of possible trees to build
- Can we find good one quickly?

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

Course ratings dataset

Greedy decision tree learning

- If I could ask one question, what question would I ask?
  - You want a feature that is most useful in predicting the rating of the course
  - A useful way of thinking about this is to look at the histogram of the labels for each feature



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
-1	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

What attribute is useful?

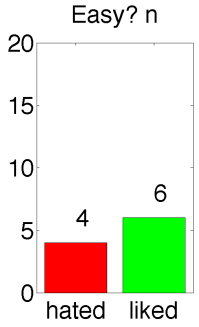
- If I could ask one question, what question would I ask?

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y				
+2	y				
+2	n				
+2	n				
+2	n				
+1	y				
+1	y				
+1	n				
0	n				
0	y				
0	n				
0	y				
-1	y				
-1	n				
-1	n				
-1	y				
-2	n				
-2	n				
-2	y				
-2	y				

Attribute = Easy?

What attribute is useful?

- If I could ask one question, what question would I ask?



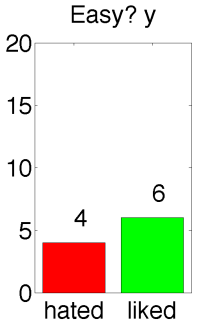
correct = 6

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	n				
+2	n				
+2	n				
+1	n				
0	n				
0	n				
-1	n				
-1	n				
-2	n				
-2	n				

Attribute = Easy?

What attribute is useful?

- If I could ask one question, what question would I ask?



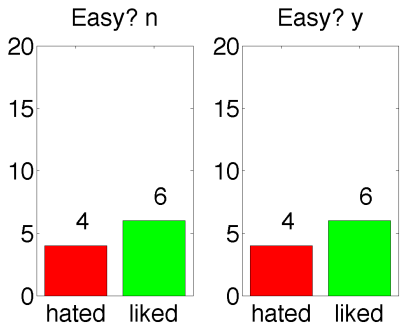
correct = 6

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y				
+2	y				
+1	y				
+1	y				
0	y				
0	y				
-1	y				
-1	y				
-2	y				
-2	y				

Attribute = Easy?

What attribute is useful?

- If I could ask one question, what question would I ask?



correct = **12**

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y				
+2	y				
+2	n				
+2	n				
+2	n				
+1	y				
+1	y				
+1	n				
0	n				
0	y				
0	n				
0	y				
-1	y				
-1	n				
-1	n				
-1	y				
-2	n				
-2	n				
-2	y				
-2	y				

Attribute = Easy?

What attribute is useful?

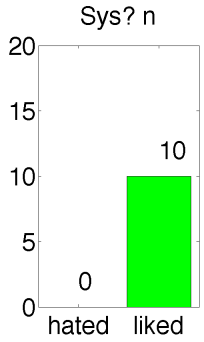
- If I could ask one question, what question would I ask?

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			n		
+2			n		
+2			n		
+2			n		
+2			y		
+1			n		
+1			n		
+1			n		
0			n		
0			n		
0			n		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

Attribute = Sys?

What attribute is useful?

- If I could ask one question, what question would I ask?



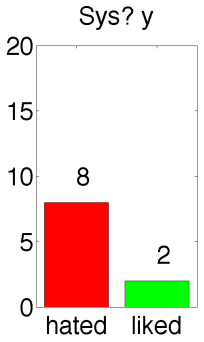
correct = 10

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			n		
+2			n		
+2			n		
+2			n		
+1			n		
+1			n		
+1			n		
0			n		
0			n		
0			n		

Attribute = Sys?

What attribute is useful?

- If I could ask one question, what question would I ask?



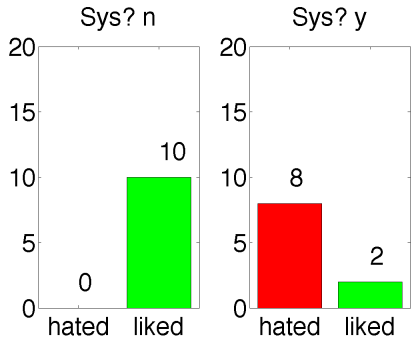
correct = 8

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			y		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

Attribute = Sys?

What attribute is useful?

- If I could ask one question, what question would I ask?

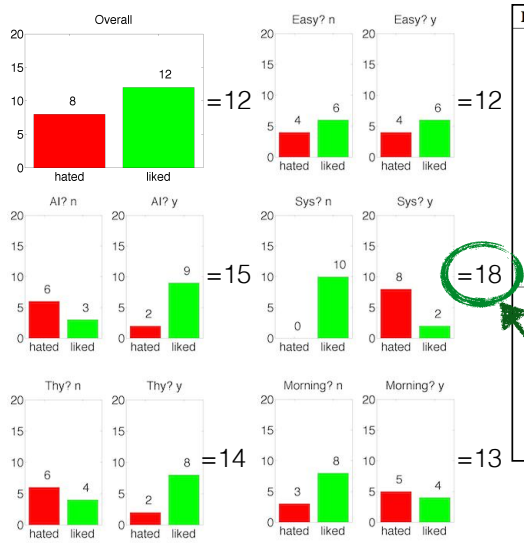


correct = **18**

Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			n		
+2			n		
+2			n		
+2			n		
+2			y		
+1			n		
+1			n		
+1			n		
0			n		
0			n		
0			n		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

Attribute = Sys?

Picking the best attribute



Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2	y	y	n	y	n
+2	y	y	n	y	n
+2	n	y	n	n	n
+2	n	n	n	y	n
+2	n	y	y	n	y
+1	y	y	n	n	n
+1	y	y	n	y	n
+1	n	y	n	y	n
0	n	n	n	n	y
0	y	n	n	y	y
0	n	y	n	y	n
0	y	y	y	y	y
-1	y	y	y	n	y
-1	n	n	y	y	n
-1	n	n	y	n	y
	y	n	y	n	y
-2	n	n	y	y	n
-2	n	y	y	n	y
-2	y	n	y	n	n
-2	y	n	y	n	y

best attribute

Decision tree training

- Training procedure
  1. Find the feature that leads to best prediction on the data
  2. Split the data into two sets {feature = Y}, {feature = N}
  3. Recurse on the two sets (Go back to Step 1)
  4. Stop when some criteria is met

- When to stop?
  - When the data is unambiguous (all the labels are the same)
  - When there are no questions remaining
  - When maximum depth is reached (e.g. limit of 20 questions)

- Testing procedure
  - Traverse down the tree to the leaf node
  - Pick the majority label

Decision tree train

Algorithm 1 DecisionTreeTrain(*data*, *remaining features*)

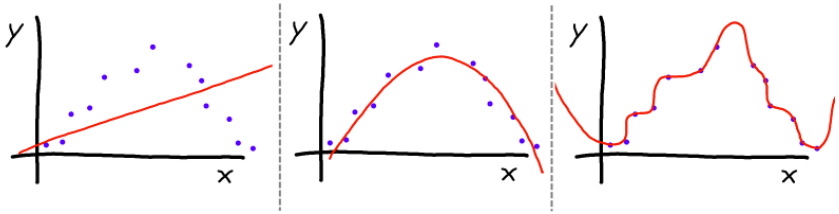
```
guess ← most frequent answer in data          // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                        // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                        // base case: cannot split further
else                                          // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                              // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
end if
```

Decision tree test

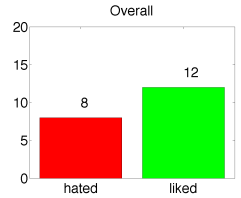
Algorithm 2 DecisionTreeTest(*tree*, *test point*)

```
1: if tree is of the form Leaf(guess) then
2:     return guess
3: else if tree is of the form Node(f, left, right) then
4:     if f = yes in test point then
5:         return DecisionTreeTest(left, test point)
6:     else
7:         return DecisionTreeTest(right, test point)
8:     end if
9: end if
```


Underfitting and overfitting



- Decision trees:
  - Underfitting: an empty decision tree
    - Test error: ?
  - Overfitting: a full decision tree
    - Test error: ?

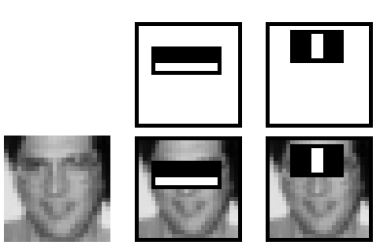


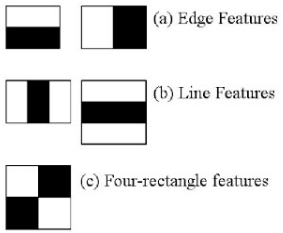
Model, parameters, and hyperparameters

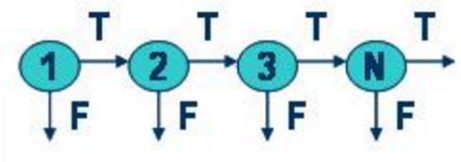
- Model: decision tree
- Parameters: learned by the algorithm
- Hyperparameter: depth of the tree to consider
  - A typical way of setting this is to use *validation* data
  - Usually set 2/3 *training* and 1/3 *testing*
    - Split the training into 1/2 *training* and 1/2 *validation*
    - Estimate optimal hyperparameters on the *validation* data

DTs in action: Face detection

- Application: Face detection [Viola & Jones, 01]
  - Features: detect light/dark rectangles in an image

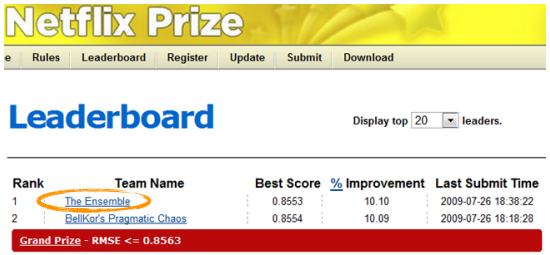






Ensembles

- Wisdom of the crowd: groups of people can often make better decisions than individuals
- Questions:
  - Ways to combine base learners into ensembles
  - We might be able to use simple learning algorithms
  - Inherent parallelism in training
  - Boosting — a method that takes classifiers that are only slightly better than chance and learns an arbitrarily good classifier

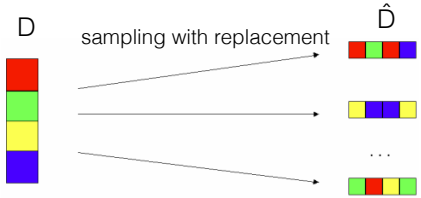


Voting multiple classifiers

- Most of the learning algorithms we saw so far are deterministic
  - If you train a decision tree multiple times on the same dataset, you will get the same tree
- Two ways of getting multiple classifiers:
  - Change the learning algorithm
    - Given a dataset (say, for classification)
    - Train several classifiers: decision tree, kNN, logistic regression, neural networks with different architectures, etc
    - Call these classifiers $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})$
    - Take majority of predictions $\hat{y} = \text{majority}(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x}))$
      - For regression use mean or median of the predictions
  - Change the dataset
    - How do we get multiple datasets?

Bagging

- Option: split the data into K pieces and train a classifier on each
  - A drawback is that each classifier is likely to perform poorly
- Bootstrap resampling is a better alternative
  - Given a dataset D sampled i.i.d from a unknown distribution $\mathcal{D}$, and we get a new dataset $\hat{D}$ by random sampling with replacement from D, then $\hat{D}$ is also an i.i.d sample from $\mathcal{D}$



There will be repetitions

Probability that the first point will not be selected:

$$\left(1 - \frac{1}{N}\right)^N \longrightarrow \frac{1}{e} \sim 0.3679$$

Roughly only **63%** of the original data will be contained in any bootstrap

- Bootstrap aggregation (bagging) of classifiers [Breiman 94]
  - Obtain datasets $D_1, D_2, \ldots, D_N$ using bootstrap resampling from D
  - Train classifiers on each dataset and average their predictions

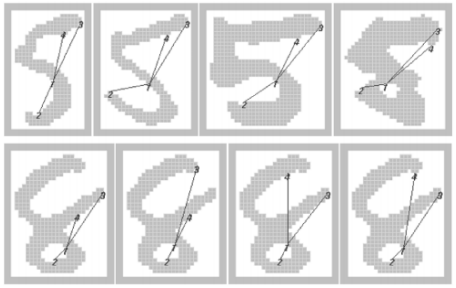
Random ensembles

- One drawback of ensemble learning is that the training time increases
  - For example when training an ensemble of decision trees the expensive step is choosing the splitting criteria
- Random forests are an efficient and surprisingly effective alternative
  - Choose trees with a fixed structure and random features
    - Instead of finding the best feature for splitting at each node, choose a random subset of size **k** and pick the best among these
    - Train decision trees of depth **d**
    - Average results from multiple randomly trained trees
  - When k=1, no training is involved — only need to record the values at the leaf nodes which is significantly faster
- Random forests tends to work better than bagging decision trees because bagging tends produce highly correlated trees — a good feature is likely to be used in all samples

DTs in action: Digits classification

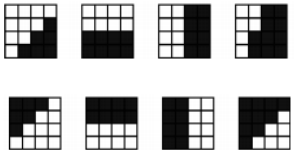
- Early proponents of random forests: "Joint Induction of Shape Features and Tree Classifiers", Amit, Geman and Wilder, PAMI 1997

Features: arrangement of tags



tags

Common 4x4 patterns



A subset of all the 62 tags

Arrangements: 8 angles

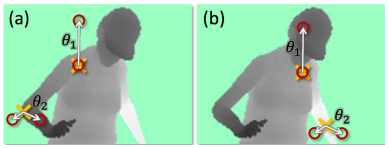
#Features: 62x62x8 = 30,752

Single tree: **7.0%** error

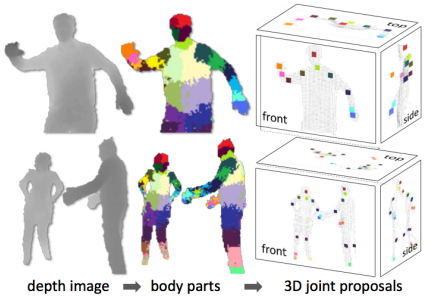
Combination of 25 trees: **0.8%** error

DT in action: Kinect pose estimation

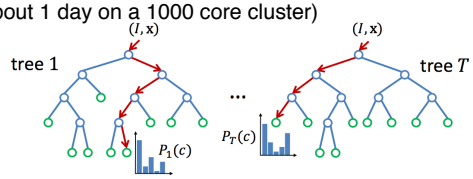
- Human pose estimation from depth in the Kinect sensor [Shotton et al. CVPR 11]



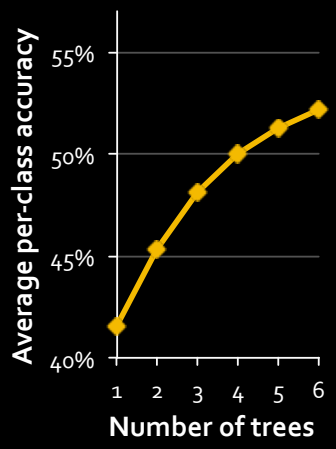
$$f_\theta(I,\mathbf{x}) = d_I\left(\mathbf{x}+\frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x}+\frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$



Training: 3 trees, 20 deep, 300k training images per tree, 2000 training example pixels per image, 2000 candidate features θ, and 50 candidate thresholds τ per feature (Takes about 1 day on a 1000 core cluster)



Number of trees



ground truth

inferred body parts (most likely)

1 tree 3 trees 6 trees

Synthetic training data

Record mocap
500k frames
distilled to 100k poses

Retarget to several models

Render (depth, body parts) pairs

Train invariance to:

Slides credit

- Decision tree learning and material are based on CIML book by Hal Daume III (http://ciml.info/dl/v0_9/ciml-v0_9-ch01.pdf)
- Bias-variance figures — https://theclevermachine.wordpress.com/tag/estimator-variance/
- Figures for random forest classifier on MNIST dataset — Amit, Geman and Wilder, PAMI 1997 — http://www.cs.berkeley.edu/~malik/cs294/amitgemanwilder97.pdf
- Figures for Kinect pose — "Real-Time Human Pose Recognition in Parts from Single Depth Images", J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, R. Moore, A. Kipman, A. Blake, CVPR 2011
- Credit for many of these slides go to Alyosha Efros, Shvetlana Lazebnik, Hal Daume III, Alex Berg, etc