CMPSCI 370: Intro. to Computer Vision
The machine learning framework

University of Massachusetts, Amherst
April 5/7, 2016

Instructor: Subhransu Maji

Administrivia

- Homework 4 due this Thursday, April 07
- Additional OH today (3-4 pm, CS 274)
- Honors section will meet today
  - We will discuss how the Microsoft Kinect works
  - Non HH students are welcome (CS 142, 4:00-5:00 pm)

Today and tomorrow

- The machine learning framework
- Common datasets in computer vision
- An example: decision tree classifiers

Classification

- How would you write a program to distinguish a picture of me from a picture of someone else?
- How would you write a program to determine whether a sentence is grammatical or not?
- How would you write a program to distinguish cancerous cells from normal cells?

Classification

- How would you write a program to distinguish a picture of me from a picture of someone else?
  - Provide examples pictures of me and pictures of other people and let a classifier learn to distinguish the two.
- How would you write a program to determine whether a sentence is grammatical or not?
  - Provide examples of grammatical and ungrammatical sentences and let a classifier learn to distinguish the two.
- How would you write a program to distinguish cancerous cells from normal cells?
  - Provide examples of cancerous and normal cells and let a classifier learn to distinguish the two.

Data (“weather” prediction)

- **Example dataset:**

Class	Outlook	Temperature	Windy?
Play	Sunny	Low	Yes
No play	Sunny	High	Yes
No play	Sunny	High	No
Play	Overcast	Low	Yes
Play	Overcast	High	No
Play	Overcast	Low	No
No play	Rainy	Low	Yes
Play	Rainy	Low	No

- **Three principal components**
  1. Class label (aka “label”, denoted by *y*)
  2. Features (aka “attributes”)
  3. Feature values (aka “attribute values”, denoted by *x*)
     - ➡ Feature values can be binary, nominal or continuous
- **A *labeled* dataset is a collection of *(x, y)* pairs**

Data (“weather” prediction)

- **Example dataset:**

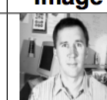
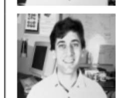
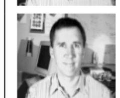
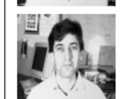
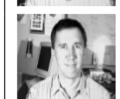
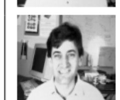
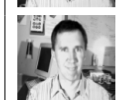
Class	Outlook	Temperature	Windy?
Play	Sunny	Low	Yes
No play	Sunny	High	Yes
No play	Sunny	High	No
Play	Overcast	Low	Yes
Play	Overcast	High	No
Play	Overcast	Low	No
No play	Rainy	Low	Yes
Play	Rainy	Low	No

- **Task:**

Class	Outlook	Temperature	Windy?
???	Sunny	Low	No

- **Predict the class of this “test” example**
- Requires us to **generalize** from the training data

Data (face recognition)

Class	Image	Class	Image
Avrim		Tom	
Avrim		Tom	
Avrim		Tom	
Avrim		Tom	

- What is a good *representation* for images? Pixel values? Edges?

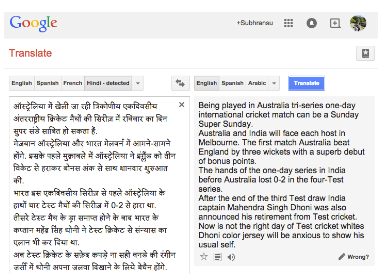
Ingredients for classification

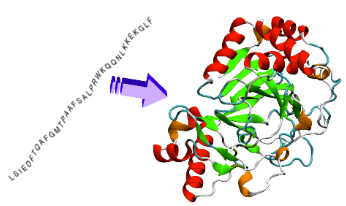
- **Whole idea:** Inject *your* knowledge into a learning system
- **Sources of knowledge:**
  1. Feature representation
     - Not typically a focus of machine learning
     - Typically seen as “problem specific”
     - However, it’s hard to learn from bad representations
  2. Training data: labeled examples
     - Often expensive to label lots of data
     - Sometimes data is available for “free”
  3. Model
     - No single learning algorithm is always good (“no free lunch”)
     - Different learning algorithms work with different ways of representing the learned classifier

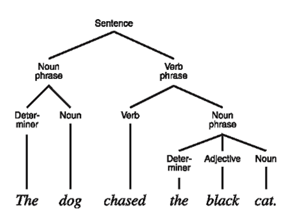
Regression

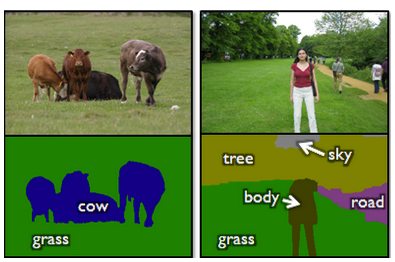
- **Regression** is like **classification** except the **labels are real valued**
- **Example applications:**
  - Stock value prediction
  - Income prediction
  - CPU power consumption
  - Your grade in CMPSCI 689

Structured prediction







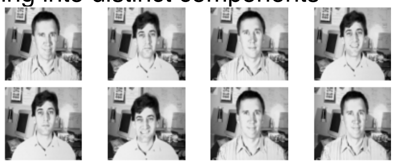


Unsupervised learning: Clustering

- **Two types of clustering**
  1. Clustering into distinct components
  2. Hierarchical clustering

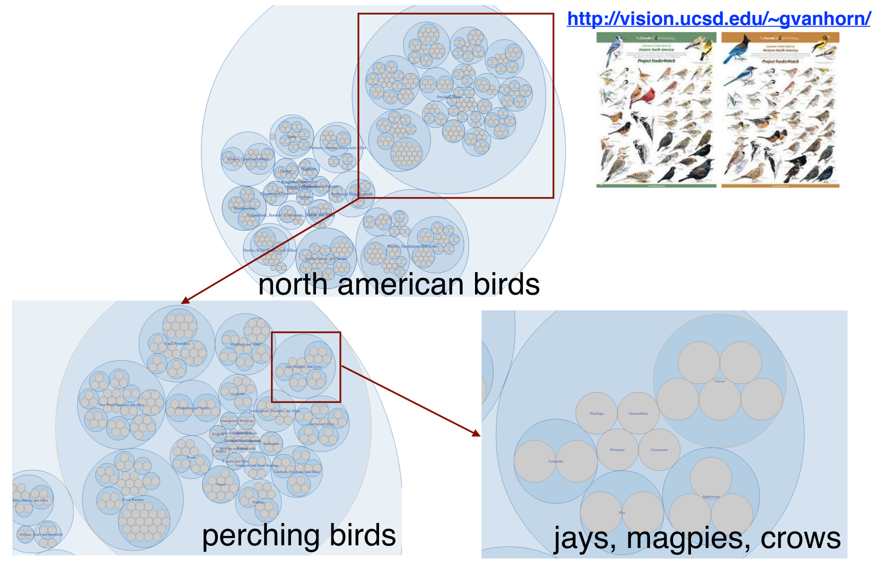
Unsupervised learning: Clustering

- **Two types of clustering**
  1. Clustering into distinct components



  - ➡ How many clusters are there?
  - ➡ What is important? Person? Expression? Lighting?
  2. Hierarchical clustering

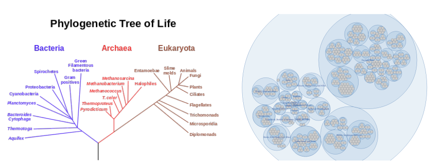
Unsupervised learning: Clustering



Unsupervised learning: Clustering

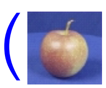
- **Two types of clustering**
  1. Clustering into distinct components

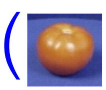
  - ➡ How many clusters are there?
  - ➡ What is important? Person? Expression? Lighting?
  2. Hierarchical clustering
  - ➡ What is important?
  - ➡ How will we use this?



Learning to recognize

Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

The machine learning framework

$$y = f(\mathbf{x})$$

output — prediction function — Image feature

Training: given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set

Testing: apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Steps

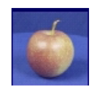
Training

Training Images

Training Labels → Training

Training Images → Image Features → Training → Learned model

Testing



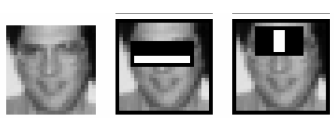
Test Image → Image Features → Prediction

Learned model → Prediction

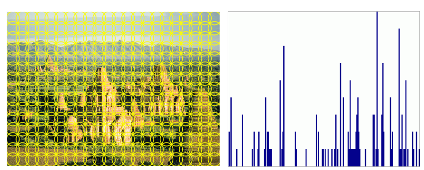
Slide credit: D. Hoiem

Features (examples)

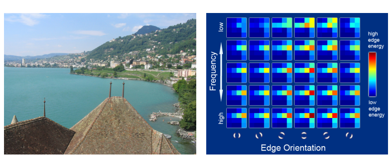
Raw pixels (and simple functions of raw pixels)



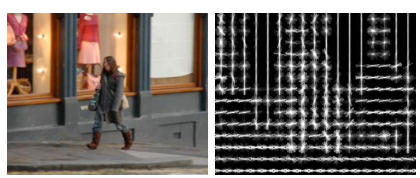
Histograms, bags of features



GIST descriptors

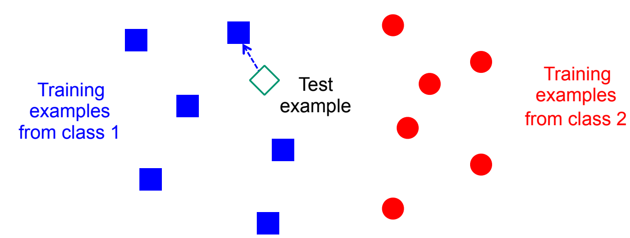


Histograms of oriented gradients(HOG)



more on this topic next week…

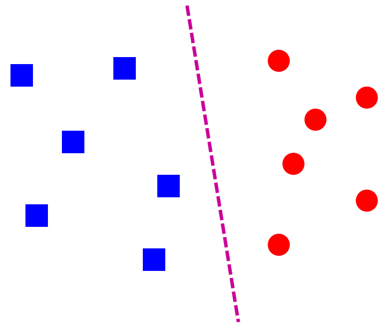
Classifiers: Nearest neighbor



$f(\mathbf{x})$ = label of the training example nearest to $\mathbf{x}$

All we need is a distance function for our inputs
No training required!

Classifiers: Linear

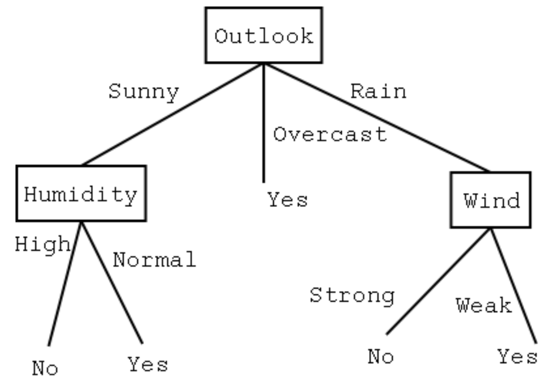


Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Classifiers: Decision trees

Play tennis?



Generalization

Training set (labels known)

Test set (labels unknown)

How well does a learned model *generalize* from the data it was trained on to a new test set?

Diagnosing generalization ability

Training error: how well does the model perform at prediction on the data on which it was trained?

Test error: how well does it perform on a never before seen test set?

Training and test error are both *high*: **underfitting**

- Model does an equally poor job on the training and the test set
- Either the training procedure is ineffective or the model is too "simple" to represent the data

Training error is *low* but test error is *high*: **overfitting**

- Model has fit irrelevant characteristics (noise) in the training data
- Model is too complex or amount of training data is insufficient

Underfitting and overfitting

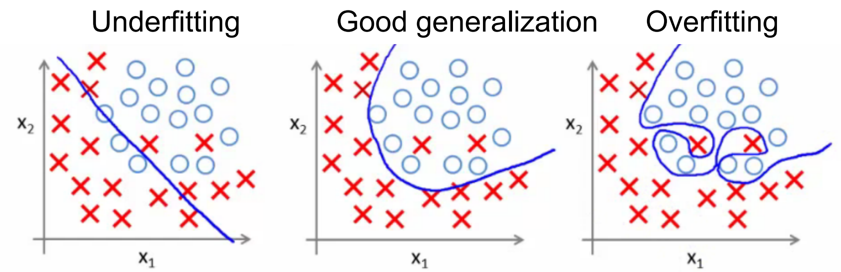


Figure source

Today and tomorrow

- The machine learning framework
- Common datasets in computer vision
- An example: decision trees

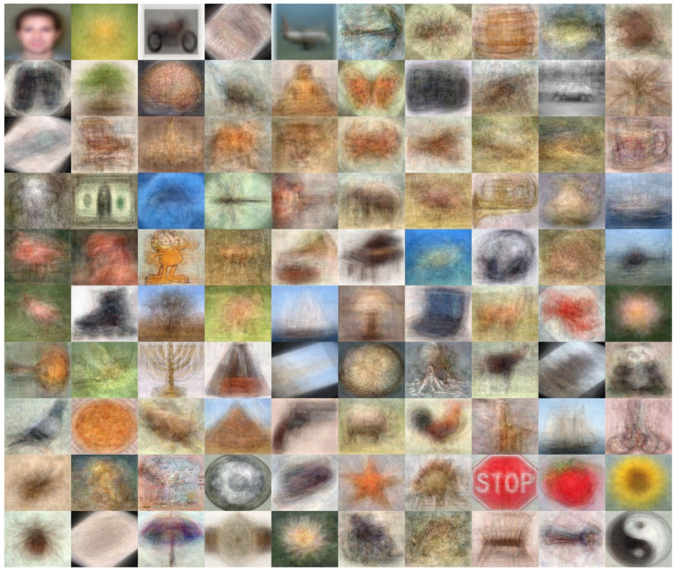
Caltech 101 & 256

http://www.vision.caltech.edu/Image_Datasets/Caltech101/
http://www.vision.caltech.edu/Image_Datasets/Caltech256/

Griffin, Holub, Perona, 2007

Fei-Fei, Fergus, Perona, 2004

Caltech-101: Intra-class variability



The PASCAL Visual Object Classes Challenge (2005-2012)

http://pascallin.ecs.soton.ac.uk/challenges/VOC/

- **Challenge classes:**
  Person: person
  Animal: bird, cat, cow, dog, horse, sheep
  Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
  Indoor: bottle, chair, dining table, potted plant, sofa, tv/ monitor

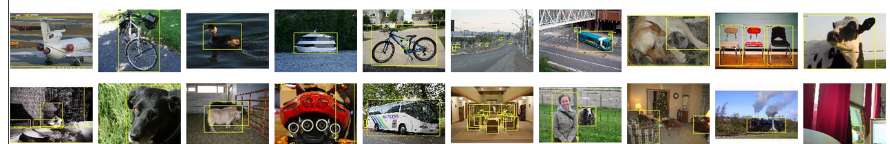
- **Dataset size (by 2012):**
  11.5K training/validation images, 27K bounding boxes, 7K segmentations

PASCAL competitions

http://pascallin.ecs.soton.ac.uk/challenges/VOC/

Classification: For each of the twenty classes, predicting presence/absence of an example of that class in the test image

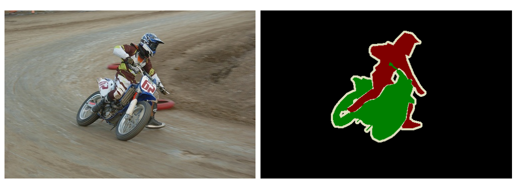
Detection: Predicting the bounding box and label of each object from the twenty target classes in the test image



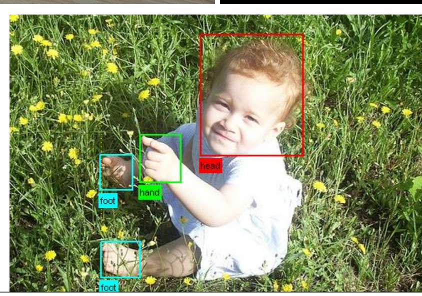
PASCAL competitions

http://pascallin.ecs.soton.ac.uk/challenges/VOC/

Segmentation: Generating pixel-wise segmentations giving the class of the object visible at each pixel, or "background" otherwise



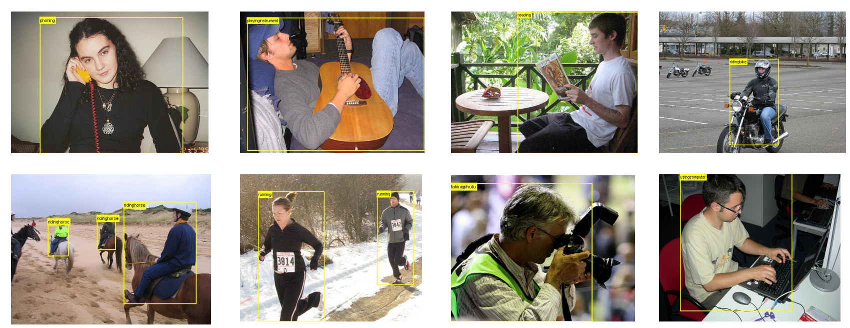
Person layout: Predicting the bounding box and label of each part of a person (head, hands, feet)



PASCAL competitions

http://pascallin.ecs.soton.ac.uk/challenges/VOC/

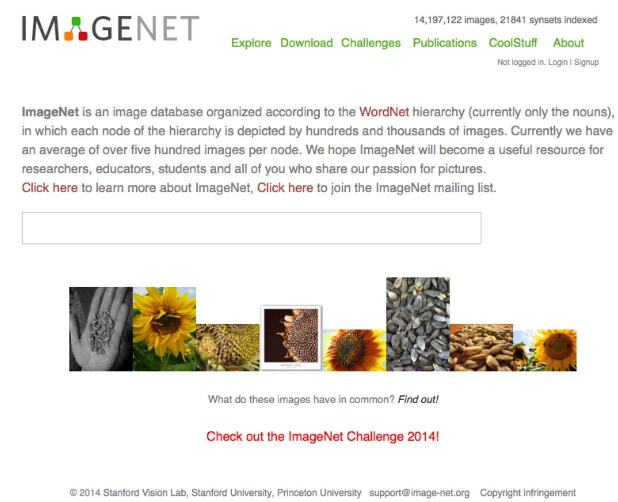
Action classification (10 action classes)



LabelMe Dataset http://labelme.csail.mit.edu/

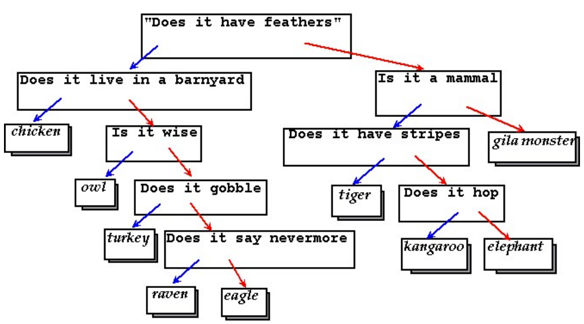
Russell, Torralba, Murphy, Freeman, 2008

ImageNet http://www.image-net.org/



Today and tomorrow

- The machine learning framework
- Common datasets in computer vision
- An example: decision tree classifiers



20 questions game

The decision tree model of learning

- Classic and natural model of learning
- Question: Will an unknown user 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

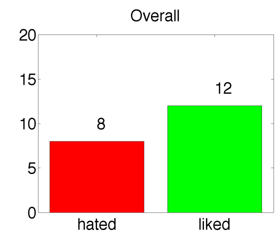
- Recall that one of the ingredients of learning is training data
  - I'll give you *(x, y)* pairs, i.e., set of (attributes, label) pairs
  - We will simplify the problem by
    - {0,+1, +2} as "liked"
    - {-1,-2} as "hated"
- Here:
  - Questions are features
  - Responses are feature values
  - Rating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

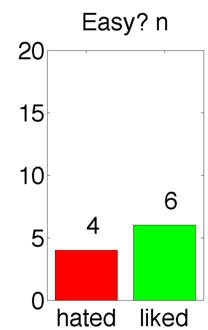
What attribute is useful?

Attribute = Easy?

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				

What attribute is useful?

Attribute = Easy?

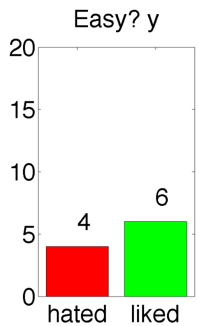


correct = 6

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				

What attribute is useful?

Attribute = Easy?

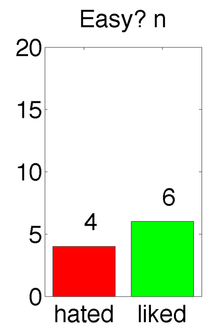


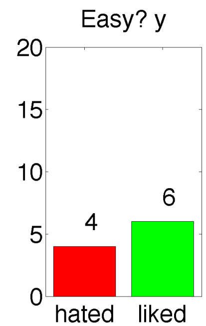
correct = 6

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				

What attribute is useful?

Attribute = Easy?





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				

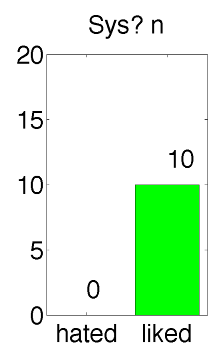
What attribute is useful?

Attribute = Sys?

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		

What attribute is useful?

Attribute = Sys?

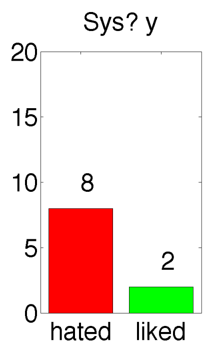


correct = 10

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		

What attribute is useful?

Attribute = Sys?

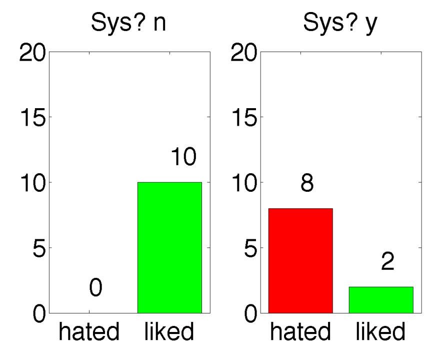


Rating	Easy?	AI?	Sys?	Thy?	Morning?
+2			y		
0			y		
-1			y		
-1			y		
-1			y		
-1			y		
-2			y		
-2			y		
-2			y		
-2			y		

correct = 8

What attribute is useful?

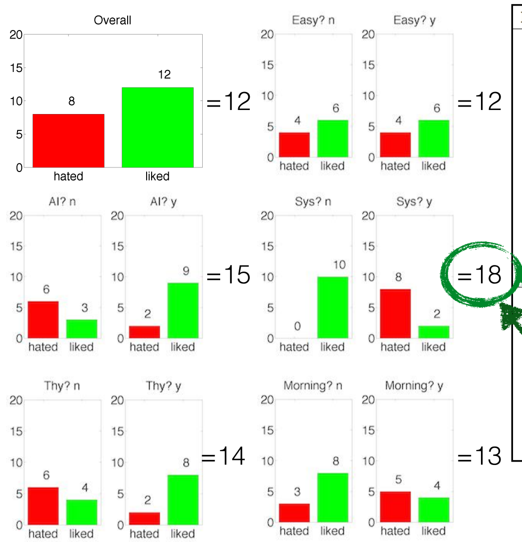
Attribute = Sys?



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		

correct = **18**

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
-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

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*)

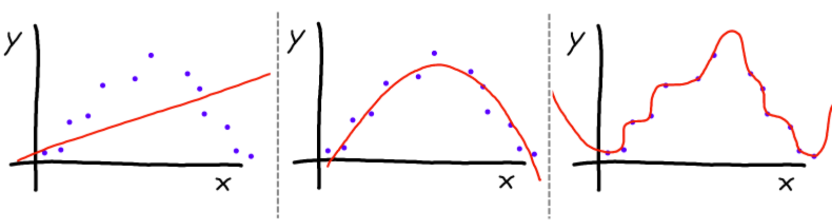
```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // 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

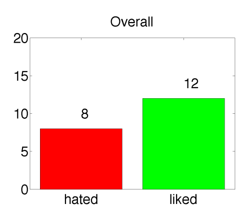
Algorithm 2 DecisionTreeTest(*tree*, *test point*)

```
if tree is of the form Leaf(guess) then
    return guess
else if tree is of the form Node(f, left, right) then
    if f = yes in test point then
        return DecisionTreeTest(left, test point)
    else
        return DecisionTreeTest(right, test point)
    end if
end if
```


Underfitting and overfitting



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

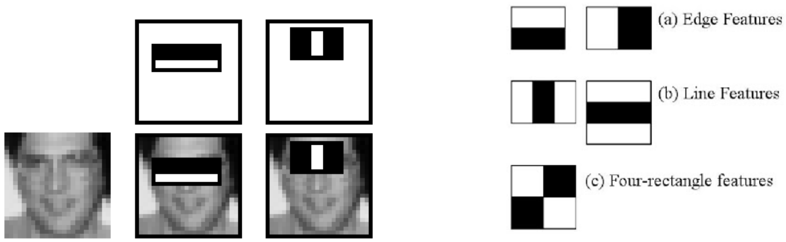


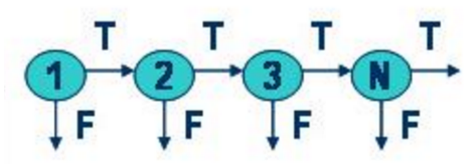
Model, parameters and hyperparameters

- 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

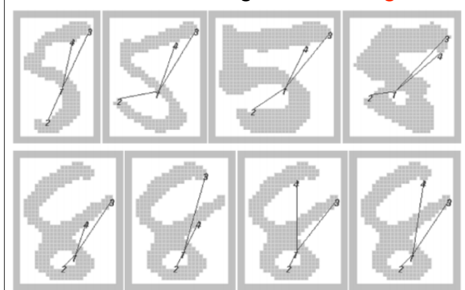




DTs in action: Digits classification

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

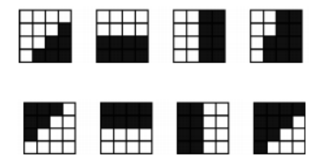
Features: arrangement of tags



Arrangements: 8 angles

tags

Common 4x4 patterns



A subset of all the 62 tags

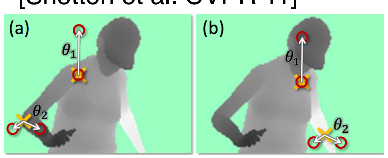
#Features: 62x62x8 = 30,752

Single tree: **7.0%** error

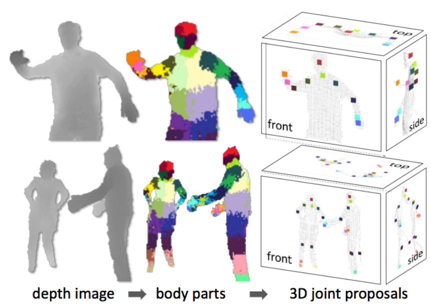
Combination of 25 trees: **0.8%** error

DT in action: Kinect pose estimation

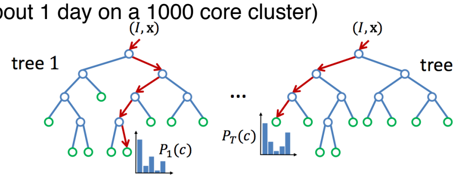
- Human pose estimation from depth in the Kinect sensor [Shotton et al. CVPR 11]



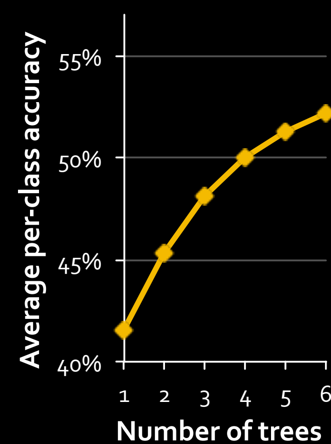
$$f_\theta(I,\mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

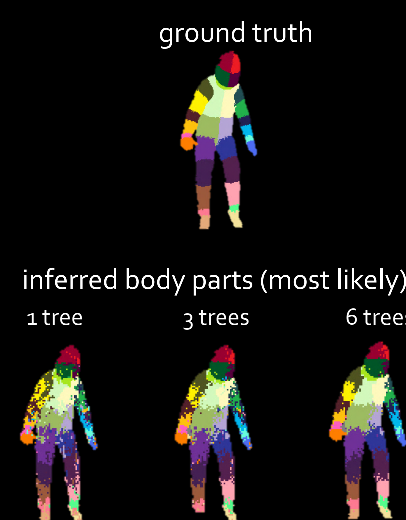


Training: 3 trees, 20 deep, 300k training images per tree, 2000 training example pixels per image, 2000 candidate features θ, and 50 candidate thresholds τ per feature (Takes about 1 day on a 1000 core cluster)



Number of trees

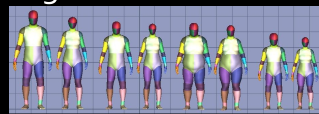




Synthetic training data

Record mocap
500k frames
distilled to 100k poses

Retarget to several models

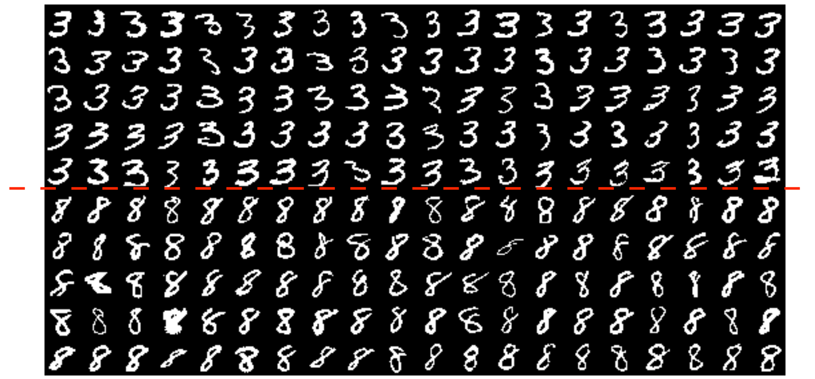


Render (depth, body parts) pairs

Train invariance to:

Homework 5

- Classify digits 3 vs. 8
  - Decision node: is pixel (x,y) is 0 or 1



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