Computer Science

# CSC380: Principles of Data Science

## Introduction to Machine Learning /

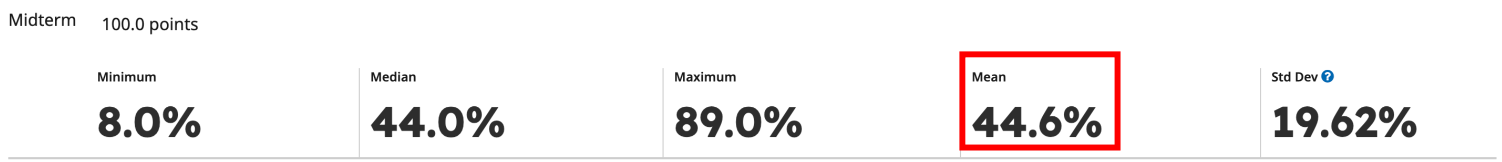
## Basics of Predictive Modeling and Classification

Kyoungseok Jang

## Announcement

- Delay: Due to the urgent circumstances of the TA in charge, we haven't finished the grading of the Midterm and HW4. We will have it done by next Tuesday. HW5 is out now (due : 3/24)
- Midterm Curving
  - I am thinking about $\sqrt{100\times(Your\ Score)}$ as the curved score.
    - E.g.) If your score is 50, your curved score is slightly over 70.

Because of the TA's circumstances, **One problem** is not graded yet.



Midterm 100.0 points

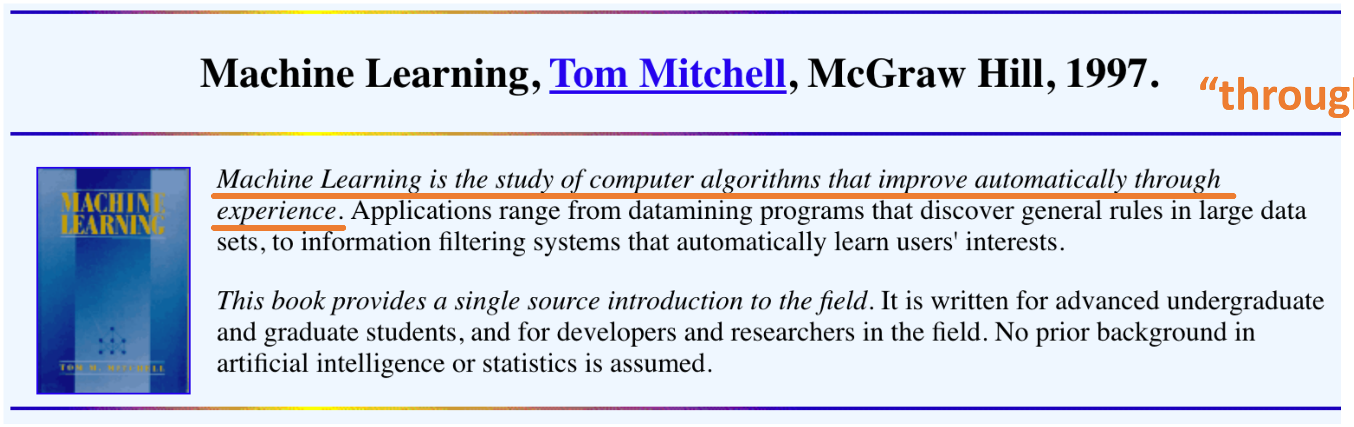
| Minimum | Median | Maximum | Mean | Std Dev |
|---|---|---|---|---|
| 8.0% | 44.0% | 89.0% | 44.6% | 19.62% |

I expect around 50% after full grading...

- Self-Withdrawal deadline: 3/28

## What is machine learning?

- **Tom Mitchell** established Machine Learning Department at CMU (2006).



**Machine Learning, Tom Mitchell, McGraw Hill, 1997.**

*Machine Learning is the study of computer algorithms that improve automatically through experience*. Applications range from datamining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests.

*This book provides a single source introduction to the field*. It is written for advanced undergraduate and graduate students, and for developers and researchers in the field. No prior background in artificial intelligence or statistics is assumed.

**"through experience"**

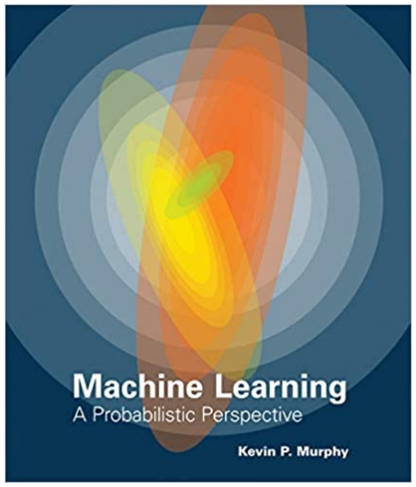
- A bit outdated with recent trends, but still has interesting discussion (and easy to read).
- A subfield of **Artificial Intelligence** – you want to perform nontrivial, smart tasks. The difference from the traditional AI is "**how**" you build a computer program to do it.

## Textbooks

*We will use a more recent textbook for readings*

*Takes a **probabilistic approach** to machine learning*

*Consistent with the goals of data science in this class*



Murphy, K. "Machine Learning: A Probabilistic Perspective." MIT press, 2012

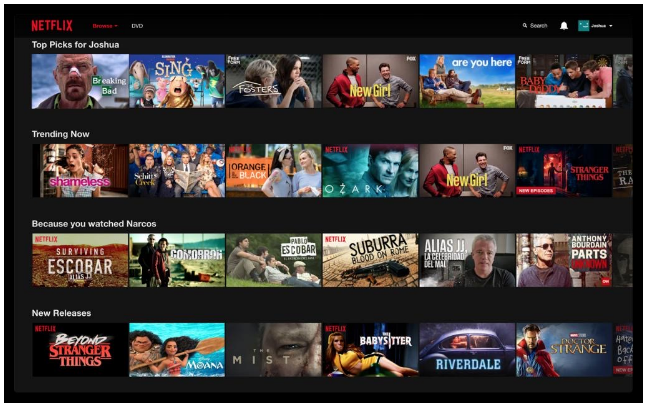
( UA Library )

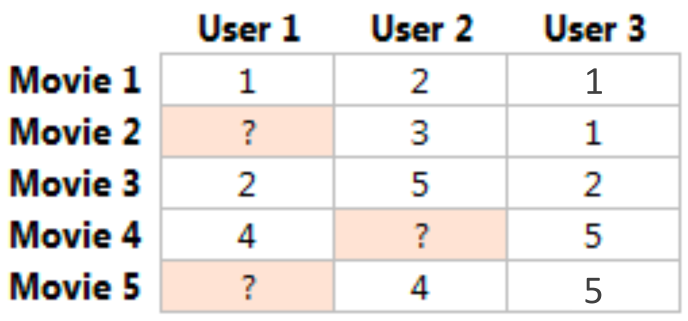
## AI Task 1: Image classification

- Predefined categories: $C$ = {cat, dog, lion, …}
- Given an image, classify it as one of the categories $c \in C$ with the highest accuracy.
- **Use**: sorting/searching images by category, medical imaging, object identification, traffic control, categorizing types of stars/events in the Universe (images taken from large surveying telescopes)

# AI Task 2: Recommender systems

- Predict how user would rate a movie
- **Use**: For each user, pick an unwatched movie with high predicted ratings. (Youtube, Netflix, Amazon, etc.)
- **Idea**: compute user-user similarity or movie-movie similarity, then compute a weighted average.



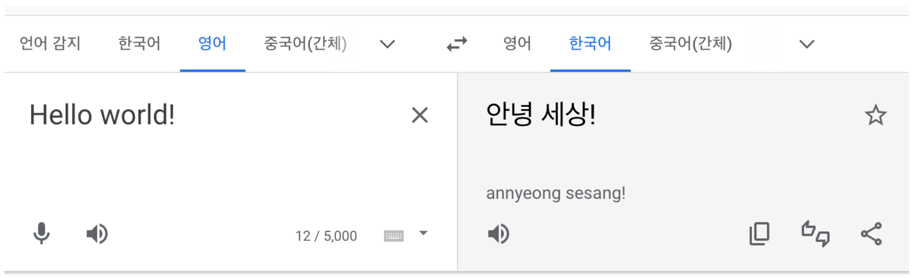


| | User 1 | User 2 | User 3 |
|---|---|---|---|
| Movie 1 | 1 | 2 | 1 |
| Movie 2 | ? | 3 | 1 |
| Movie 3 | 2 | 5 | 2 |
| Movie 4 | 4 | ? | 5 |
| Movie 5 | ? | 4 | 5 |

"collaborative filtering"

# AI Task 3: Machine translation

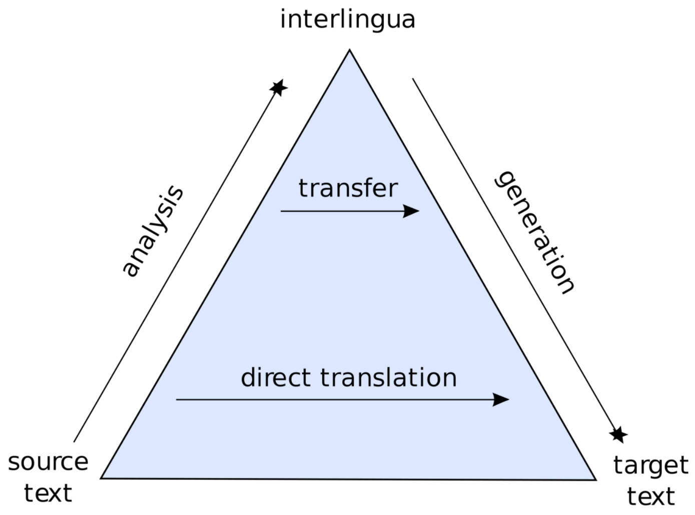
- No need to explain how useful it is.
- **<u>Task</u>**: 1) Transform a sentence to the interlingual language (analysis) and 2) create a sentence with another language with the same meaning, with appropriate grammar structure (generation).



Hello world!

안녕 세상!

annyeong sesang!



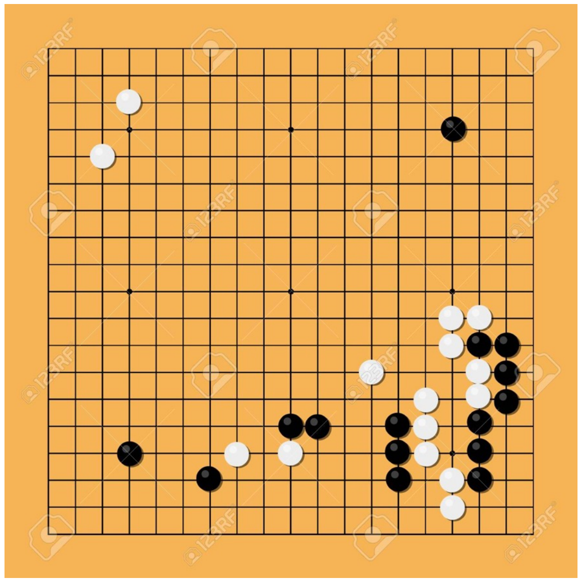
# AI Task 4: Board game

- Predict win probability of a move in a given game state (e.g., AlphaGo)
- Traditionally considered as a "very smart" task to perform.

Q: how will it be useful for us, though?

- **Use**: From the AI Go player, you can do practice play or even learn from it.
  - Now it's a major trend in the field of Go

- **Potential use**: Board game (e.g., Catan) design, better AI
- Deeply related to robot AI and autonomous driving
  - Predict the future of your move



## Traditional AI vs Machine Learning (ML)

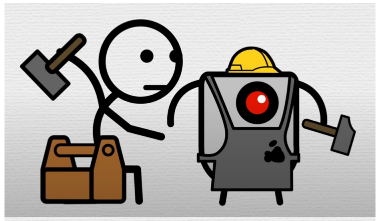
- **Traditional AI**: *you* encode the knowledge (e.g., logic statements/rules), and the *machine* executes it.
  - e.g., if there is feather-like texture with two eyes and a beak, classify it as a bird.
  - Advancements in automated '**inference**' like "if a -> b and b-> c, then a-> c". => 'expert system'
- **ML**: Given a set of input and output pairs (e.g., animal picture + label), and train a **function** (a set of logical statements / a neural network) that maps the input to the output accurately.
  - As the "big data" era comes, data is abundant => turns out, better than systems based on hand-coded domain knowledge!
  - "statistical" approach // data-driven approach

*"Every time I fire a linguist, the performance of the speech recognizer goes up."*
– *1988,* Frederick Jelinek, a Czech-American researcher in information theory & speech recognition.

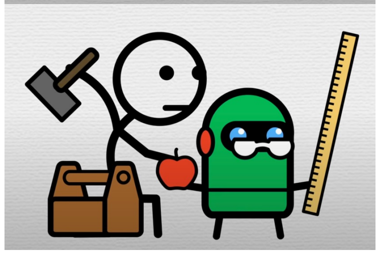
# Traditional AI vs Machine Learning (ML)

- Traditional AI – watchmaker
  - You encode your knowledge (springs and parts) directly
  - You understand why those parts are necessary.

- Machine Learning (ML) – one example (from https://www.youtube.com/watch?v=R9OHn5ZF4Uo)

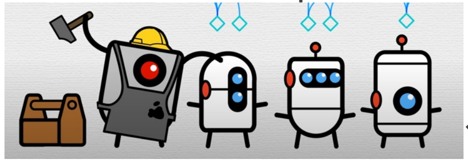


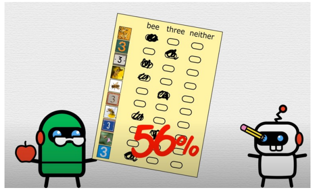
Auto-updater



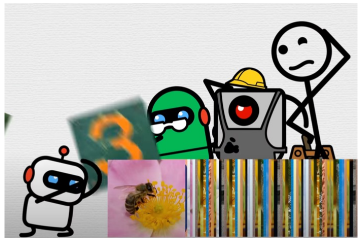
Auto-tester

Automatically improve through Interaction and experience





It is possible that you might not know why your function is designed in that way



## Overview of ML Methods

**Supervised Learning**

- Provide *training* data consisting of input-output pairs and learn mapping
- E.g., Spam prediction, object detection or image classification, machine translation, etc.

**Unsupervised learning**

- **No predefined categories**. Finds patterns in the data without the help of labels (outputs)
- E.g., clustering, dimensionality reduction, target tracking, image segmentation, etc.

**Reinforcement learning** ⇐ **We won’t cover this**

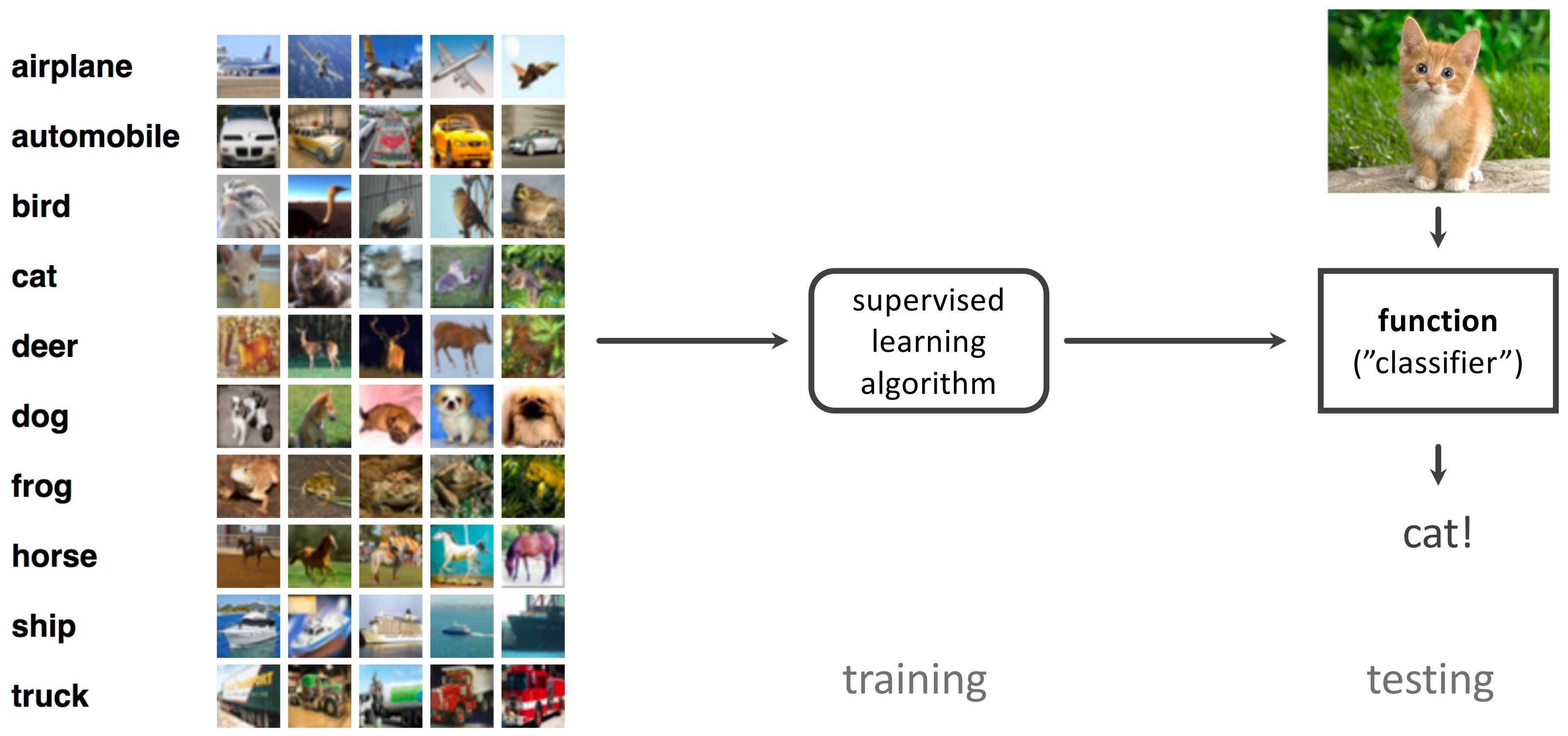
- The environment interacts with your action, transferring you to different states.
- E.g., autonomous driving, robot AI, recommendation system

## Supervised Learning

## Basic setting: Supervised learning

example = data point
labeled = categorized

- Train data: dataset comprised of *labeled examples*: a pair of (input, label)



## Example function 1: Decision tree

```
Task: predict the 5-star rating of a movie by a user
If age >= 60 then
   if genre = western then
      return 4.3
   else if release date > 1998 then
      return 2.5
   else ...
   ...
   end if
else if age < 60 then
...
end if
```

training:

- determine the shape of the tree
- which condition to have at each node
- what to output from each leaf node

## Example function 2: Linear

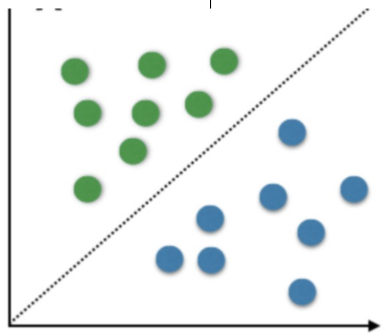
```
Task: Image classification

Let $x$ be a set of pixel values of a picture (30x30) =>
900-dimensional vector $x$.

If $\boxed{0.124 \cdot x_1 - 2.5 \cdot x_2 + \cdots + 2.31 \cdot x_{900} - 2.12} \geq 0$ then
   return cat
else
   return dog
end
```

called feature vector

"linear combination"/"inner product"

training:

- determine the coefficients & threshold

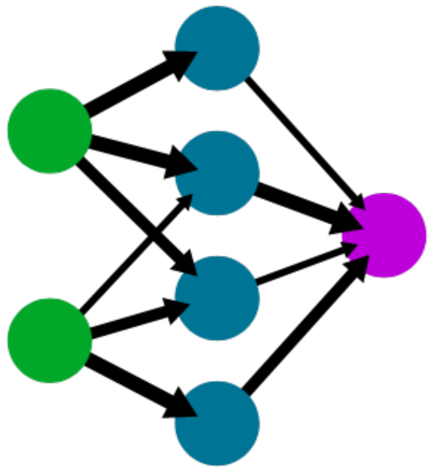


E.g., in 2d space, it induces a linear decision boundary:

$0.5 \cdot x_1 - 2.5 \cdot x_2 > 4.3$

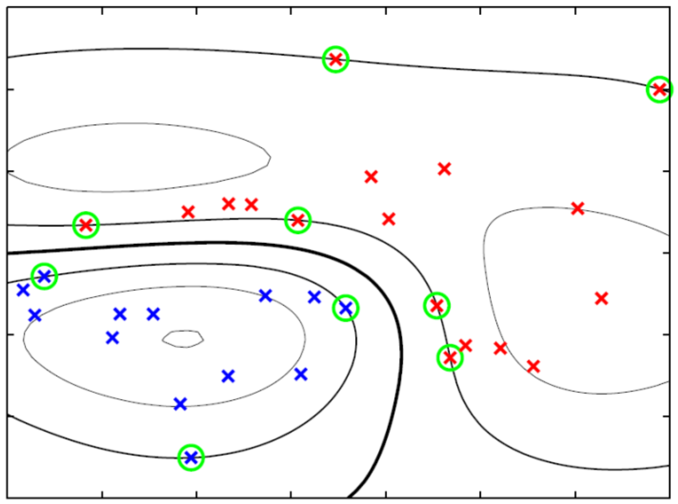
## Example function 3: Nonlinear

Neural network



(stacked **linear** models with nonlinear **activation functions**)

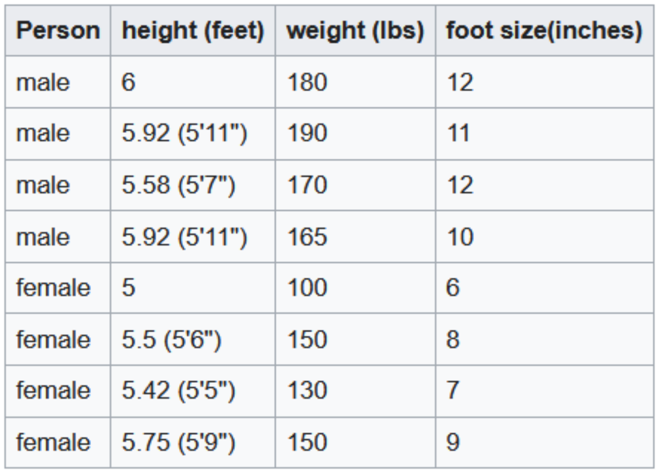
Support Vector Machine



(**linear** in the induced feature space)

# Example: Naïve Bayes Classifier

**Training Data:**



| Person | height (feet) | weight (lbs) | foot size(inches) |
|---|---|---|---|
| male | 6 | 180 | 12 |
| male | 5.92 (5'11") | 190 | 11 |
| male | 5.58 (5'7") | 170 | 12 |
| male | 5.92 (5'11") | 165 | 10 |
| female | 5 | 100 | 6 |
| female | 5.5 (5'6") | 150 | 8 |
| female | 5.42 (5'5") | 130 | 7 |
| female | 5.75 (5'9") | 150 | 9 |

Features

**Task:** Observe feature vector $x = (x_1, \dots, x_n)$ and predict class label $y \in \{1, \dots, C\}$

**Model:** Treat features as *conditionally independent*, given class label:

$$p(x, y) = p(y) \prod_{i=1}^{n} p(x_i | y)$$

Doesn't capture correlation among features, but is easier to learn.

**Classification:** Bayesian model so classify by posterior,

$$p(y = c \,|x) = \frac{p(C = k)p(x|y = c)}{p(x)}$$

## Supervised learning: Types of prediction problems

**Binary classification:** Choose between 2 classes

- Given an email, is it spam or not? (or the probability of it being a spam)

**Multi-class classification:** more than 2 categories.

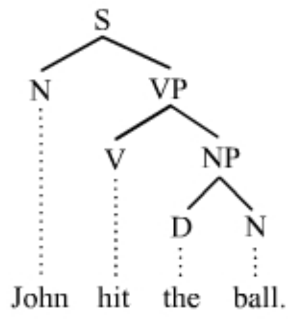
- Image classification with 1000 categories. (cat, dog, airplane, car, computer, …)

**Regression:** the label is real-valued (e.g., price)

- Say I am going to visit Italy next month. Given the price trends in the past, what would be the price given (the # of days before the departure, day of week)?
- Predict the stocks/bitcoin price in the future

**Structured output prediction:** more than just a number

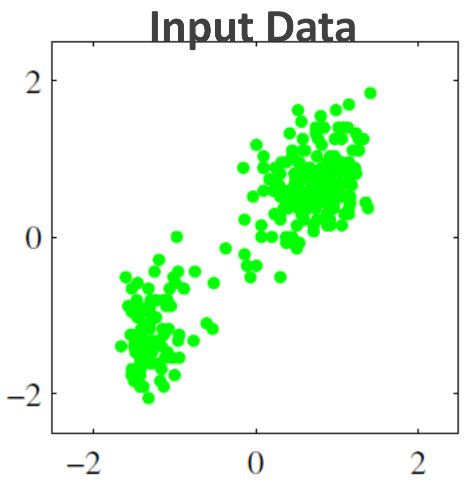
- Given a sentence, what is its grammatical parse tree?

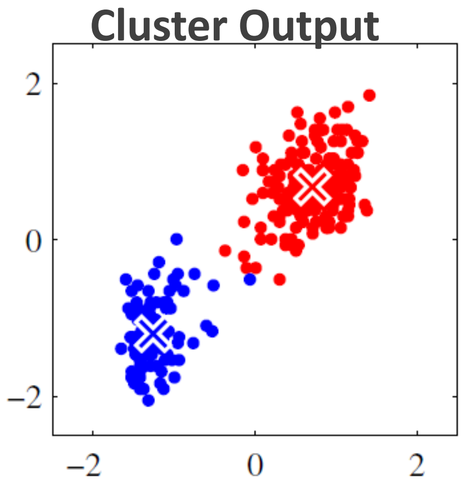


## Unsupervised Learning

## Example: Clustering

Identify groups (clusters) of similar data





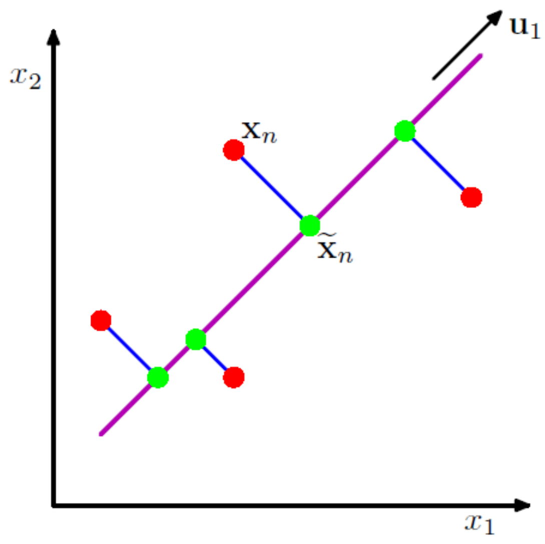
Useful for interpreting large datasets

Clusters are assigned arbitrary labels (e.g. 1, 2, …, K).
=> afterwards, you may look at the data and name each group.

Common clustering algorithms: K-means, Expectation Maximization (EM)

## Example: Principal Component Analysis (PCA)

Reduce dimension of high-dimensional data using linear projection



Identify directions of **maximum variation** in the data by computing ***eigenvectors***

*Easier explanation: Identify important directions*

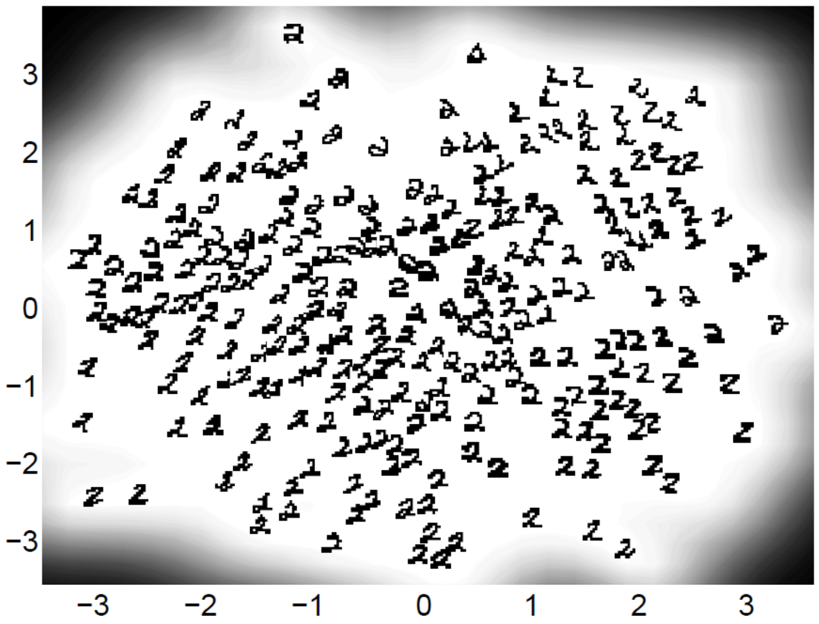
Linear projection onto K-dimensional subspace spanned by top K eigenvalues

Can be used for visualization (project to 2D) or for compressing images.

**Source: Bishop, C. PRML**

## Example: Principal Component Analysis (PCA)

Reduce dimension of high-dimensional data using linear projection

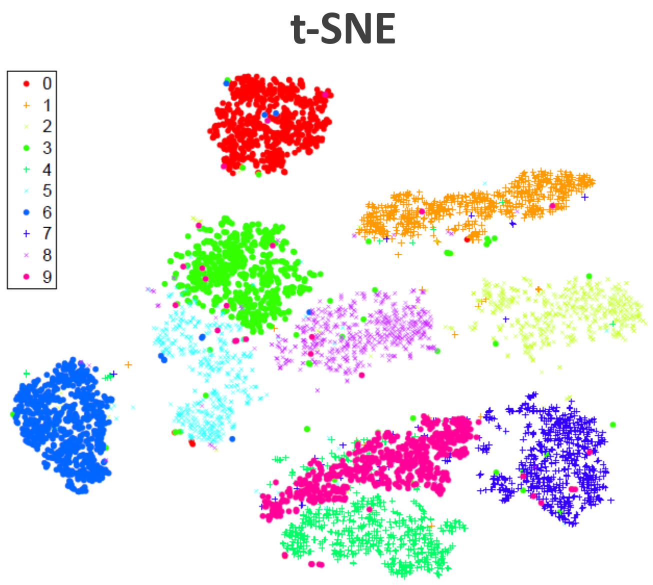


**Source: Lawrence, N. (2005)**

Example for modeling / visualizing handwritten digits

Each digit is a black/white image with 28x28 pixels (784 dimensions) projected down to 2D

## Example: Nonlinear Dimensionality Reduction



Nonlinear reduction can (potentially) amplify clustering properties

**t-Distributed Stochastic Neighbor Embedding (t-SNE)** Models similarity between data as a t distribution and strives to find projection that preserves similarity.

## Example: Generative models

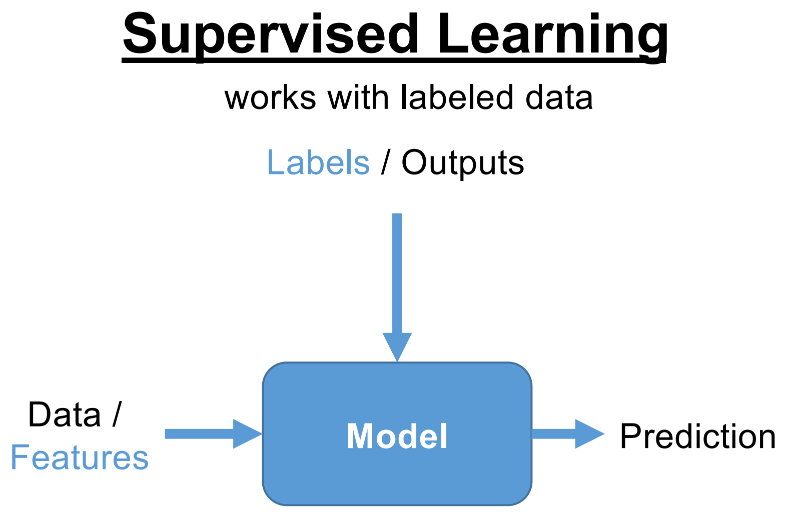
**We won't cover this**

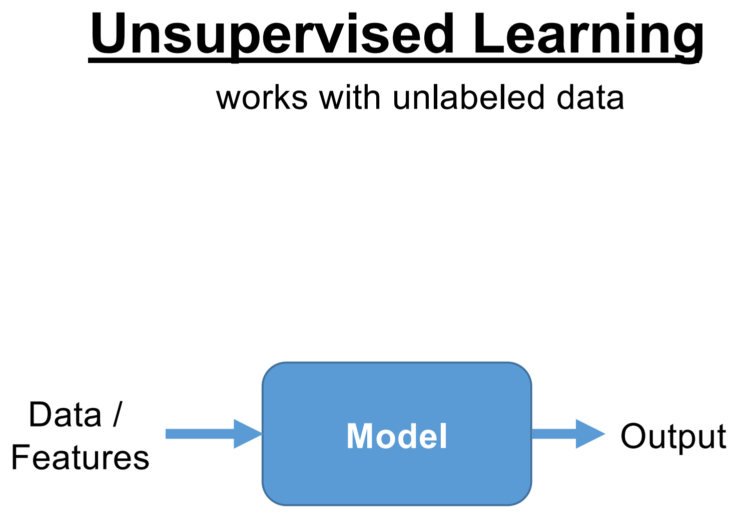
- AI image generators
- It is hard to define how 'good' the generated image is.
  - How can we explain the 'painting style' to computers? Mostly impossible… → Unsupervised!

## Summary

- **Supervised Learning** - Training data consist of inputs and outputs
  - Classification, regression, translation, …
- **Unsupervised Learning** – Training data only contain inputs
  - Clustering, dimensionality reduction, segmentation, …
- **Linear** models generate output as a *linear combination* of inputs,
  - E.g. $y = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d$
  - PCA, linear regression, etc.
- **Nonlinear** models fit an arbitrary nonlinear function to map inputs-outputs
  - Neural networks, support vector machine, nonlinear dimensionality reduction

## Training Machine Learning Models





**ML models distinguished by a number of factors**

- Number of parameters needed (parametric / nonparametric)
- Whether they model uncertainty (probabilistic / nonprababilistic)
- Do they model the data generation process? (generative / discriminative)

Computer Science

# CSC380: Principles of Data Science

## Basics of Predictive Modeling and Classification 1: Decision Tree

Kyoungseok Jang

## Decision Trees

figures/examples from “A Course in Machine Learning” by Hal Daume III  http://ciml.info/

## Majority Vote Classifier

The most basic classifier you can think of.

How to train:

- Given: A (train) dataset with m data points $\{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ with C classes.
- Compute the most common class $c^*$ in the dataset.

$$c^* = \arg\max_{c \in \{1,\ldots,C\}} \sum_{i=1}^{m} \mathbf{I}\{y^{(i)} = c\}$$

- Output a classifier $f(x) = c^*$.

Example:
Data: m=10
$x^{(i)}$: images of cats and dogs
$y^{(i)}$: label (cat/dog)
Suppose that there are 6 dogs and 4 cats.
After 'training', your classifier always outputs 'dog', even without looking at the input.

Stupid enough classifier! Always try to beat this classifier.

Often, state-of-the-art ML algorithms perform barely better than the majority vote classifier..
$\Rightarrow$ happens when there is no association between features and labels in the dataset

## Train set accuracy

- Suppose the ML algorithm has trained a function $f$ using the dataset $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ where $x^{(i)}$ is input and $y^{(i)}$ is label.
- Train set accuracy:

$$\widehat{acc}(f) := \frac{1}{m} \sum_{i=1}^{m} \mathbf{I}\{f(x^{(i)}) = y^{(i)}\}$$

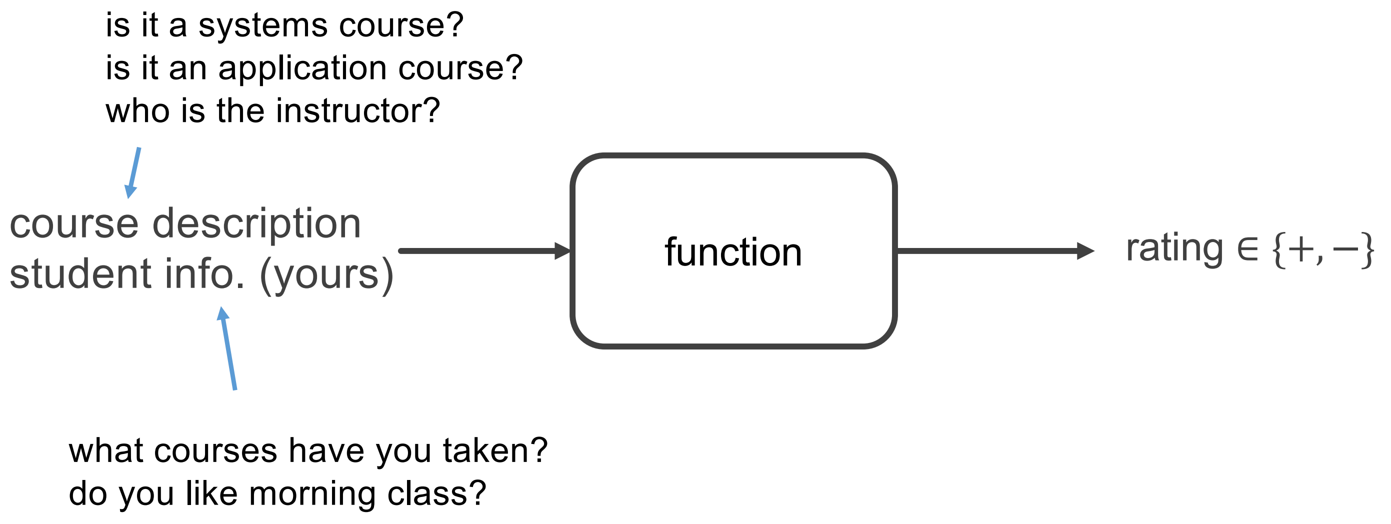
It is the number of times the function got the answer right divided by m.

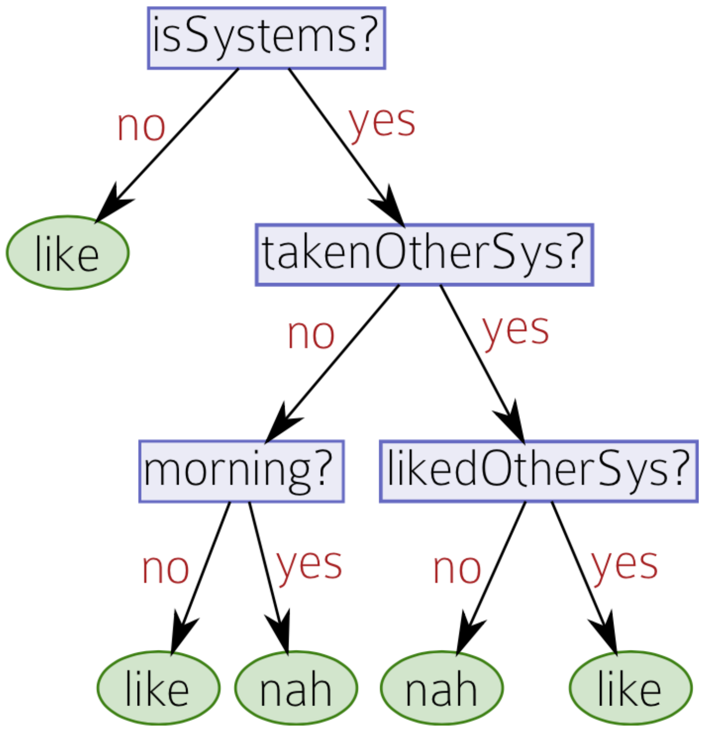
- Q: We have 100 data points (images) with 5 cats, 80 dogs, and 15 lions. What is the train set accuracy of the majority vote classifier?

.80

## Decision tree (example: course recommendation)

- Build software: recommend a set of courses for you
  - More precisely, given a course, predict its rating





Wouldn’t it be nice to construct such a tree automatically by a computer algorithm?

Wouldn’t it be nice if it accurately predicts?

You can, if you have data!

HasTakenPrereqs (=: Prereq)

HasTakenACourseFromTheSameLecturer (=: Lecturer)

HasLabs

| 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 |

consider it to be 'like'

consider it to be 'dislike'

For example, this table is data D.
Each row is a course you've rated.
$x^{(i)}$ is a sequence of 5 yes/no (d=5) for i-th course.
$y^{(i)}$ is the sign of the rating for i-th course.

Define the data $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$

$\in \{y, n\}^d$

$\in \{+, -\}$

Each dimension of $x^{(i)}$ is called a **feature**.
$x^{(i)}$ is called a **feature vector**.

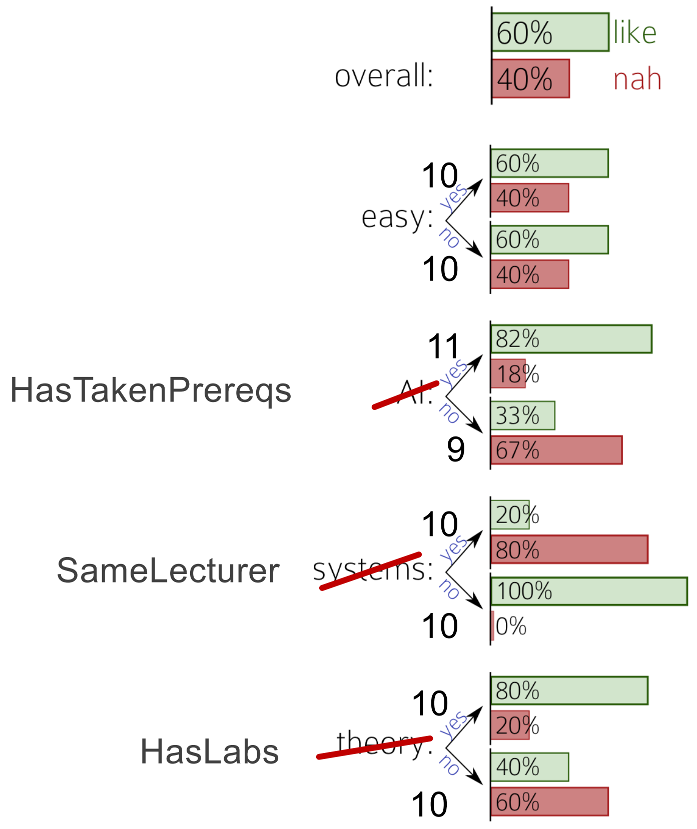
## How to Train a Tree

- Main principle: Find a tree that has a high train set accuracy

$$\widehat{acc}(f) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{I}\{f(x^{(i)}) = y^{(i)}\}$$

- This is essentially the main principle governing pretty much all the machine learning algorithms!
  - "Empirical risk minimization" principle
    (empirical risk := 1 – train_accuracy)

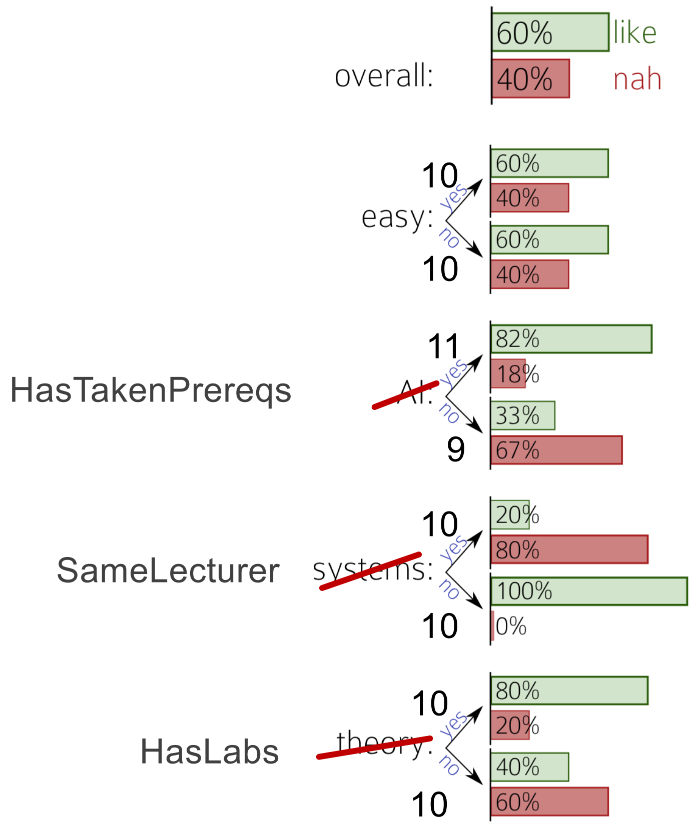
# How to construct a tree



Prereqs Lecturer HasLabs

| 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 |

## How to construct a tree

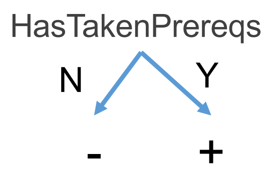


Baseline: majority vote classifer

Q: What is the train set accuracy? 0.60

Major (+)

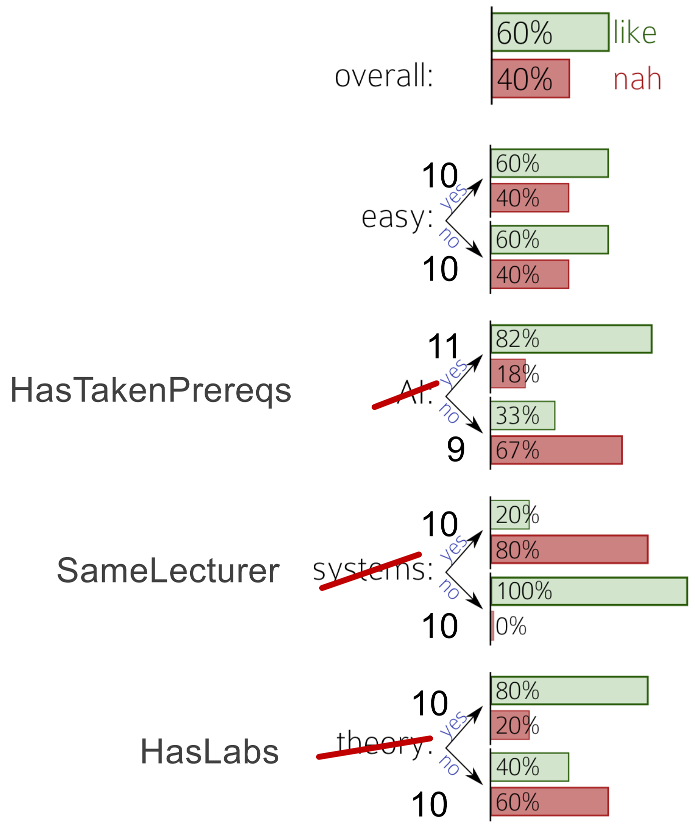
Suppose we place the node HasTakenPrereqs at the root.
Set the prediction at each leaf node as the majority vote.



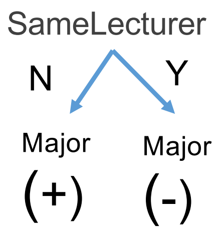
What is the train set accuracy now?

$$\frac{9}{20}\cdot\frac{6}{9}+\frac{11}{20}\cdot\frac{9}{11}=\frac{15}{20}=0.75$$ improved!

# How to construct a tree



Suppose placing the node SameLecturer at the root.



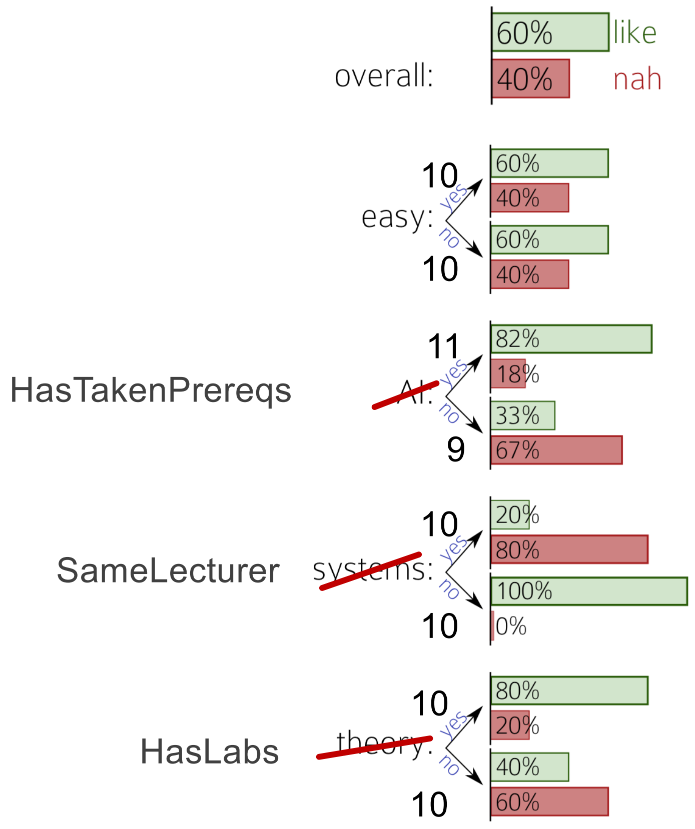
What is the train set accuracy now?

$$\frac{10}{20}\cdot\frac{10}{10}+\frac{10}{20}\cdot\frac{8}{10}=\frac{18}{20}=0.9$$ even better!

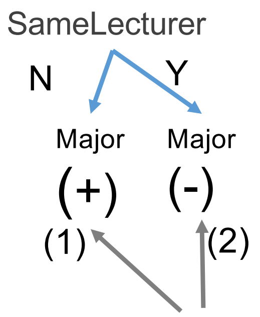
What would you do to build a depth-1 tree?

try out each feature and choose the one that leads to the largest accuracy!

# How to construct a tree



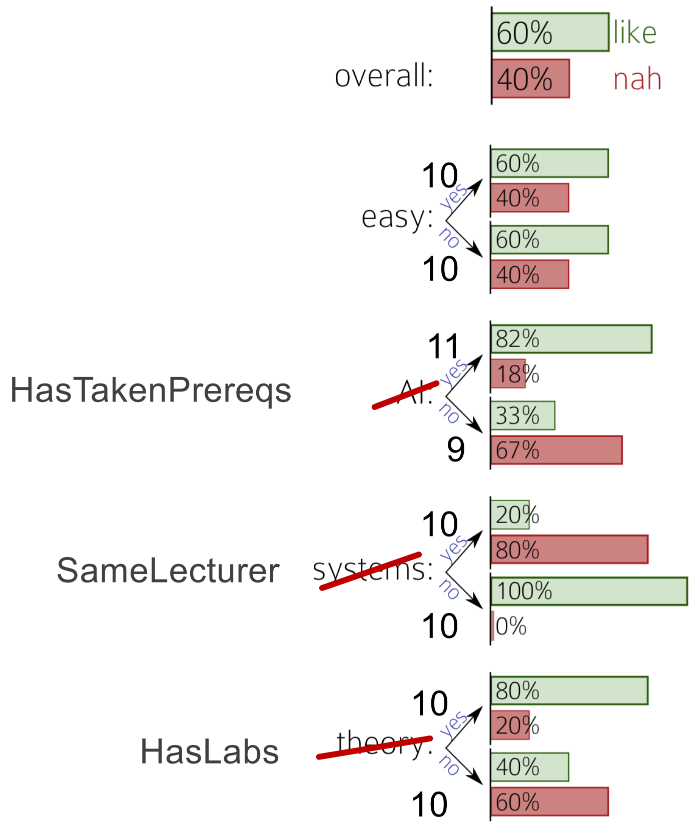
What about depth 2?

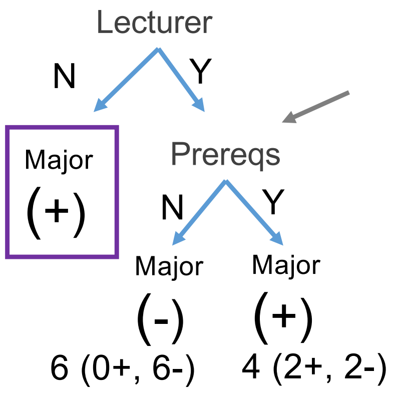


Which nodes to put at each leaf node?

Focus on (2). Try placing HasTakenPrereqs

# How to construct a tree





Q: How many training data points fall here? 10

Q: How many training data points arrive at these two leaves? How many for each label?

Q: what prediction should we use for each leaf?

Q: What is the train set accuracy, conditioning on SameLecturer=Y?

'local' train set accuracy

$$\frac{6}{10} \cdot \frac{6}{6} + \frac{4}{10} \cdot \frac{2}{4} = \frac{8}{10}$$

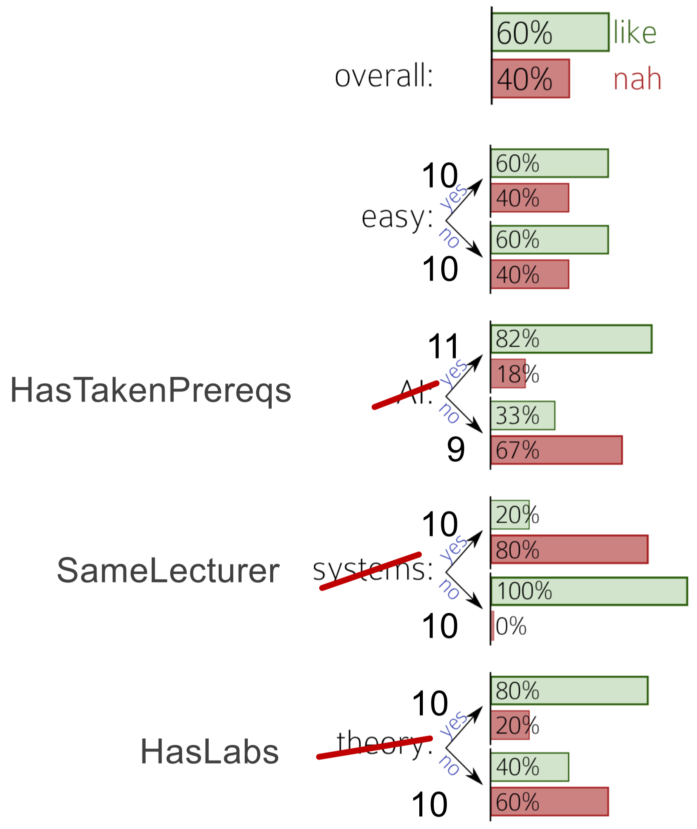
Try all the other nodes and pick the one with the largest acc.!

Then, repeat the same for SameLecturer=N branch!

=> but this has 1 local train set acc. So leave it be!

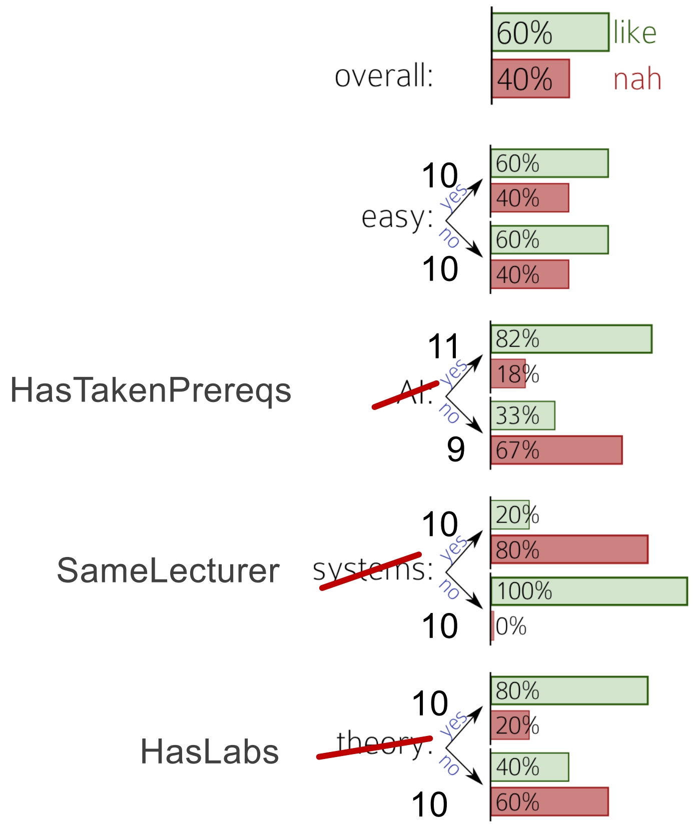
Move onto expanding nodes at depth 2!

# How to construct a tree



|  |  | Prereqs | Lecturer | HasLabs |  |
|---|---|---|---|---|---|
| **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 |

# How to construct a tree



Overall idea:

1. Set the root node as a leaf node.
2. Grab a leaf node for which its 'local' train accuracy is not 1.
3. Find a feature that maximizes the 'local' train accuracy and replace the leaf node with a node with that feature; add leaf nodes and set their predictions by majority vote.
4. Repeat 2-3.

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

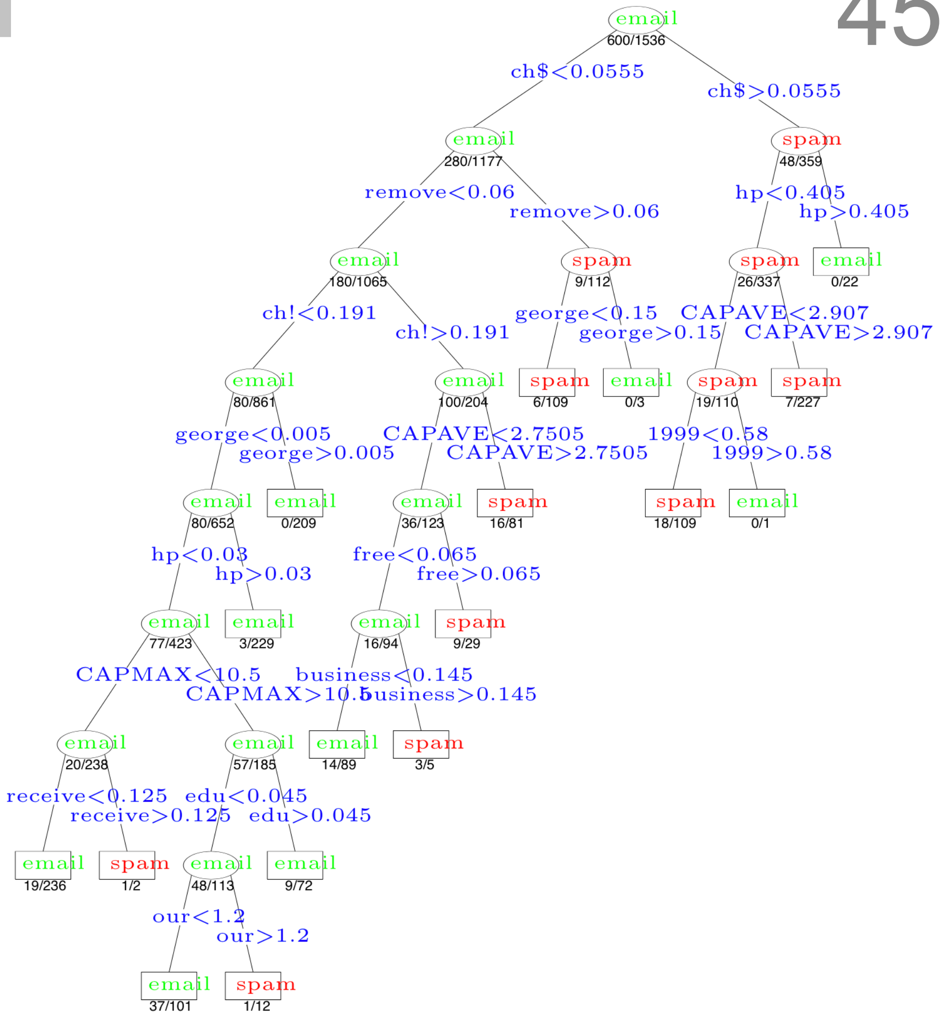
```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then              <= i.e., all data points have the same label
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all $f \in$ remaining features do               <= there is no point in adding a feature
                                                           that appeared in its parent!
        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                <= answer = label
                       + # of majority vote answers in YES ) / size(data)
    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 $\setminus \{f\}$)
    right ← DecisionTreeTrain(YES, remaining features $\setminus \{f\}$)
    return Node($f$, left, right)
end if
```

**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 $f = no$ in test point then
        return DecisionTreeTest(left, test point)
    else
        return DecisionTreeTest(right, test point)
    end if
end if
```

## Example: spam filtering I

- Spam dataset
- 4601 email messages, about 39% are spam
- Classify message by spam and not-spam
- 57 features
  - 48 are of the form “percentage of email words that is (WORD)”
  - 6 are of the form “percentage of email characters is (CHAR)”
  - 3 other features (e.g., “longest sequence of all-caps”)
- Final tree after pruning has 17 leaves, 9.3% test error rate



## Background: Train Error vs Test Error

Error := 1 – accuracy.

Suppose we have trained a function $\hat{f}$ on $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ using a supervised learning algorithm.

- Train error: Evaluate on D.

$$\widehat{err}_D(f) := \frac{1}{|D|} \sum_{(x,y)\in D} \mathbf{I}\{f(x) \neq y\}$$

- Test error: Evaluate on $D' = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m'}$ not used for training.
  - It can be possible that our function just 'memorized' the training data and doesn't do well in real life. (overfitting)

Q: Choose one:
(1) train error $\geq$ test error (2) train error $\approx$ test error (3) train error $\leq$ test error

## Background: Workflow of Training a Classifier

Standard practice:

- Given a data set D, split it into train set $D_{train}$ and $D_{test}$
  - large data: 90-10 ratio
  - medium data: 80-20 ratio
  - small data: 70-30 ratio

(these are guidelines only)

- Train on $D_{train}$ and evaluate error rate on $D_{test}$. You trust that $D_{test}$ will be the performance when you deploy the trained classifier.

Discussion: What would be reasonable logics behind such a trust?

Thank you!