Hot topics, round 2

...

~scharf~

News flash

- U.S. Gov't is subpoena-ing major crypto exchanges (link)
- Amazon, Berkshire, JPMorgan to create healthcare company (link)
- Facebook is banning ads promoting cryptocurrencies (link)
- U.S. to probe Apple over older phone slowdown (link)
- Waymo launches a fleet (link)

<big text>

Machine Learning

AI

Deep Learning

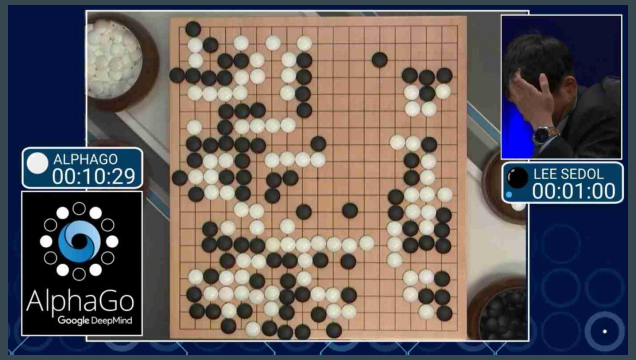
</big text>

Overview

- “A field of computer science that gives computers the ability to learn without being explicitly programmed.” -Wikipedia
  - Really just a bunch of fancy math involving iterative optimization
- Highly quantitative, usually requires a PhD
  - But *extremely* lucrative once you get into industry
- Like web programming, has applications in almost any industry.

Brief History

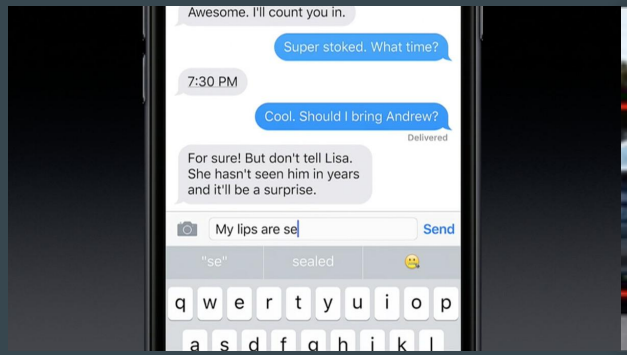
- 1950: Alan Turing proposes a 'learning machine' that could learn and become artificially intelligent, and “Turing Test”
- 1951: Arthur Samuel of IBM develops checkers-playing machine.
- 1967: Innovation of Nearest Neighbors Algorithm
- 1970s - 1980s: “AI Winter”
- 1997: IBM’s Deep Blue beats the world champion at chess
- 2016: Google’s AlphaGo beats Go world champion

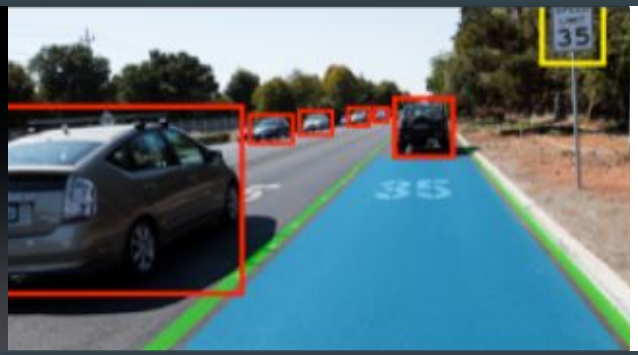


- Recent advances in GPUs

Applications

- Natural Language Processing
- Image Recognition
- Recommendation Systems
- Strategy Optimization
- *Virtually any regression or classification problem*





So how does it all work?

- Data:
  - Training Set: A set of data points, usually labelled with “features” and “truth values”, that helps you train your algorithm.
  - Test Set: A set of data points used solely to test the performance of your algorithm.
- Objective: Optimize the weighting applied to each feature such that the predicted values are as similar to the truth values as possible.
  - Overfitting: When your predictions skew heavily to the training data and fails to generalize.

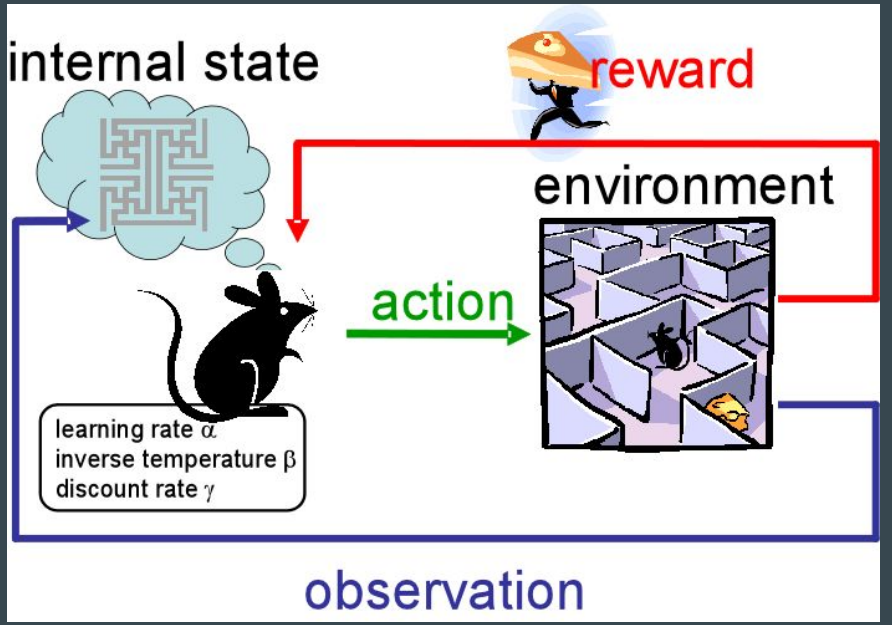
Does Yuki want to play tennis today?

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Demo #1

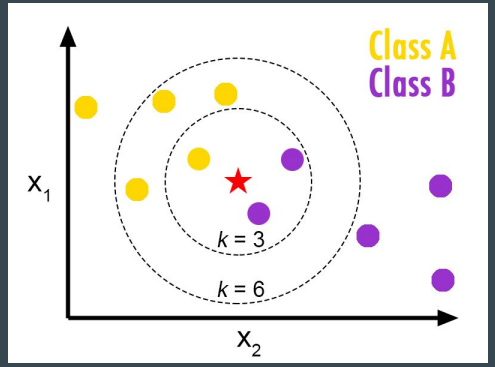
3 Main Types of ML Algorithms

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



K-Nearest Neighbors

1. Create an N-dimensional space, where each dimension represents a feature
2. Place each datapoint into the space, making sure to label them with truth values
3. For each test datapoint:
    a. Find its location in the space
    b. Calculate its Euclidean Distance to every other datapoint
    c. Classify it with the value that is shared the most amongst the K closest points



PROS

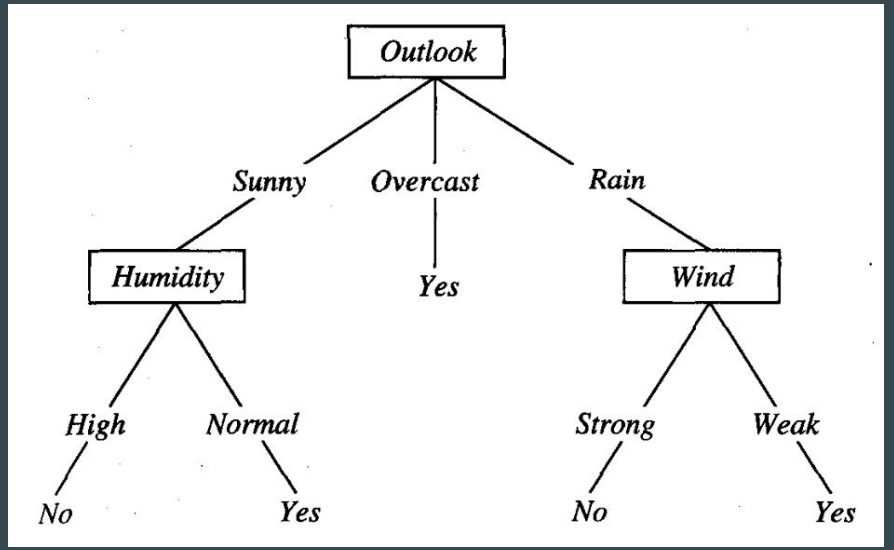
- Simple, yet powerful
- No training involved (“lazy”)
- Naturally handles multiclass classification

CONS

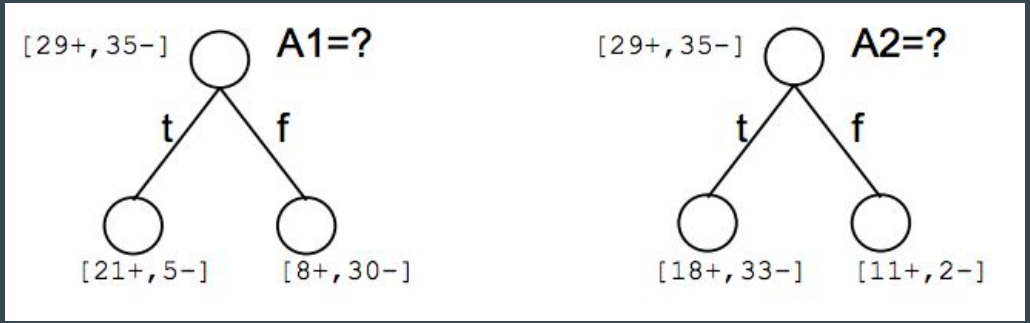
- Expensive and slow to predict
    - *Especially* on high-dimensional data
- Fails to handle poorly-sampled/noisy data

Decision Trees

- Structure:
  - Nodes are features
  - Branches are values of features
  - Leaves are classes



1. Choose the “best feature” for current node
   a. Best feature determined by entropy reduction
2. Label that feature on node, and create all appropriate branches.
3. Repeat by creating new nodes at end of each new branch, unless perfectly classified at that point.



PROS

- Fast at classifying
- Robust to noise and missing values

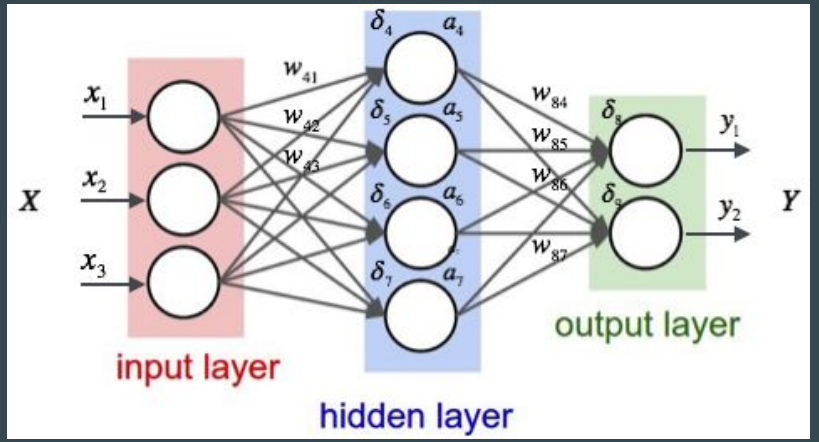
CONS

- Involves pre-processing
- Complex trees are hard to interpret

Demo #2

Neural Networks

- Easily the most complicated classifier
  - Although having just one “layer” is equivalent to a simple linear regression problem.
- Given some input to the current node, apply some weight vector to each feature and produce output, which is “thresholded”



Demo #3

Convolutional Neural Networks

131	162	232	84	91	207
104	-1	0	+1	237	109
243	-2	0	+2	135	126
185	-1	0	+1	61	225
157	124	25	14	102	108
5	155	116	218	232	249

Resources

- AI Experiments: AI project demos and source code
- Kaggle: Big AI community, has many nice datasets, forums, and competitions
- COMP135 Spring ‘16 Repo: Hosts slides and assignments of a Tufts ML class that was taught in spring 2016
- Y U K I ! ! ! !

Operating Systems & Embedded Devices

Getting close to the hardware

What does an operating system do?

- Abstraction of hardware (kernel)
  - Applications can be written for a specific operating system rather than a specific piece of hardware
  - Imagine needing a special version of your browser for each network card manufacturer!
- Sharing of hardware (also kernel)
  - In general, hardware can only do one thing at once
  - Operating systems allow multiple tasks to run concurrently and use the same resources
  - General purpose vs real-time operating systems
- Implementation of common functionality (libraries)
  - E.g. memory management, windowing systems, cryptographic libraries

Types of computing devices

- General-purpose (e.g. desktops, laptops, phones)
  - Designed for all computing tasks
  - Runs a general purpose OS
  - Can run third-party software by design
- Appliance (e.g. Wi-Fi router, Amazon Echo, Google Chromecast)
  - Designed for one computing task
  - Usually runs a general purpose OS with vendor-specific drivers or libraries
  - Only runs first-party software by design or has locked-down API
- Embedded (e.g. microwave, refrigerator, stereo system, automobile)
  - Used as part of a larger product whose primary purpose is not computing
  - Generally runs a real-time OS or no OS at all (“bare metal”)

Writing OSes & bare metal applications

- Work directly with the underlying hardware
  - Want to send a network packet? Better get out the network card's datasheet
- Common functionality isn't provided
  - "Hello world" isn't so easy when you first need to write font rendering code
  - No such thing as `new`/`malloc()` —if you want some memory, find it yourself!

What makes this fun?

- Fewer layers of abstraction to deal with
- Gives an unparalleled understanding of what's actually happening
- Allows you to make hardware do things it never could before
  - Remember, bare metal code doesn't have to be from scratch!
  - By writing a new device driver for Linux, you can make that device usable by every existing applications at once

Demo

Resources

- COMP 40 (Machine Structure and Assembly Language Programming)
- COMP 50CP (Concurrent Programming)
- COMP 111 (Operating Systems)
- COMP 140 (Advanced Computer Architecture)
- MITRE Embedded CTF (eCTF)
  - Attack/defend competition
  - Runs annually in spring
- Open-source projects
  - Linux is a fully open-source kernel. It's not hard to work on!
  - Many third-party OSes for appliances exist (OpenWRT, OpenELEC, pfSense). Porting one of these to a new device can teach you a lot and serves a purpose!