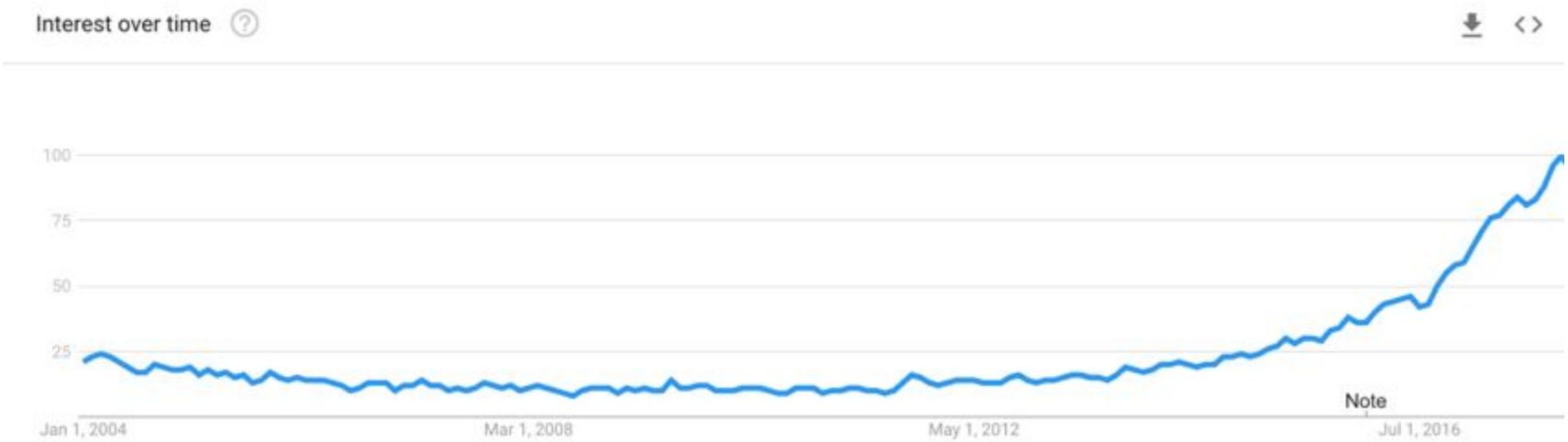
# Intro Lecture

## CS1420: Machine Learning

Michael L. Littman
Spring 2018

# Interest in Machine Learning via Google

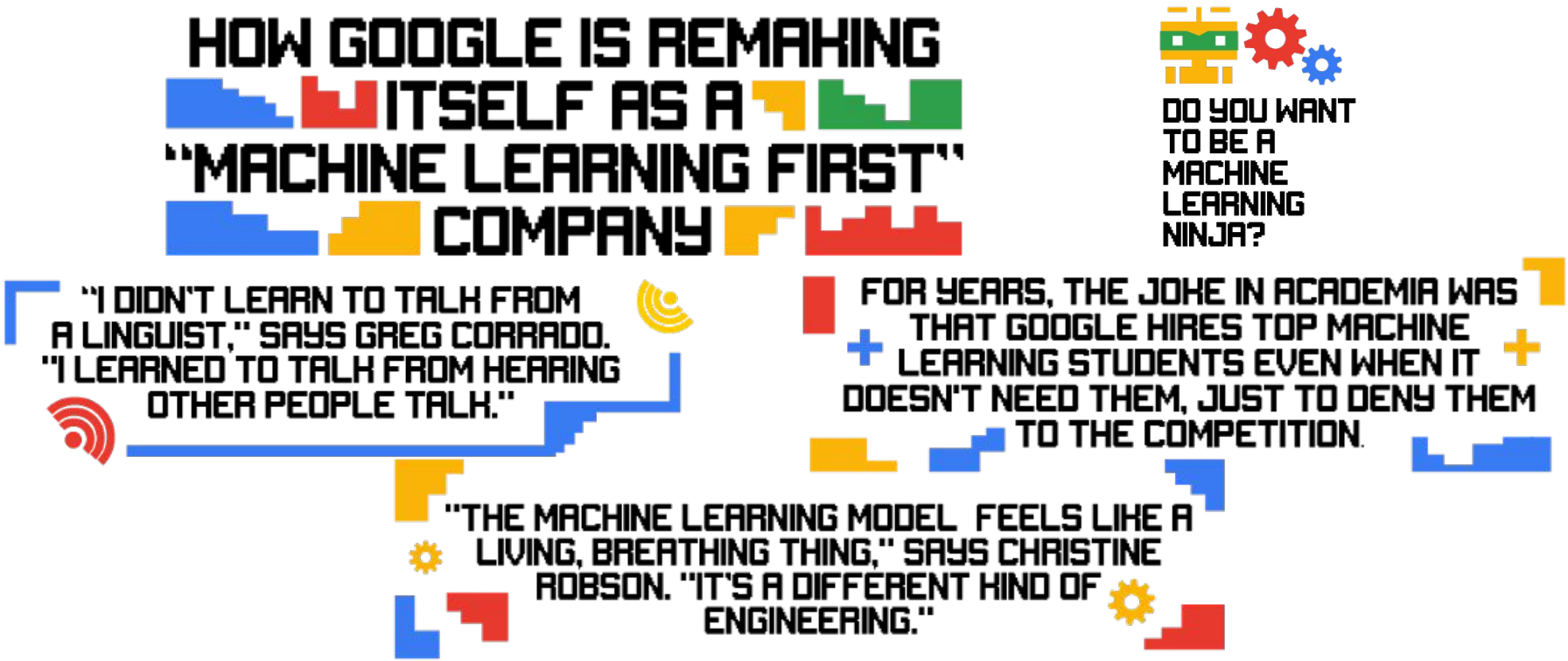
Google trends, “Machine Learning (field of study)”.




https://trends.google.com/trends/explore?date=all&q=%2Fm%2F01hyh_

# Interest in Machine Learning at Google

https://www.wired.com/2016/06/how-google-is-remaking-itself-as-a-machine-learning-first-company/



# Meet the TAs

Sam Saarien (ssaarine) [grad]
Joshua Chipman (jchipman)  [HTA]
Sean Segal (ss97) [HTA]
Kyle Lin (kjlin)
Nathaniel Brennan (nbrennan)
Sahakait Benyasut (sbenyasu)
Shihang Zhang (szhang96)
T Alexander Chen (tc59)
Vivek Ramanujan (vramanuj)
Xi Chen (xc13)
Yazen Alani (yalani)
Yue Guo (yg5)
Yu Xiang (yxiang3)
Andrew Hou (zhou)
Dan Murphy (dmurphy)

# Meet the Instructor

Michael Littman (mlittman) [instructor]

CS faculty here for the past 5 years.

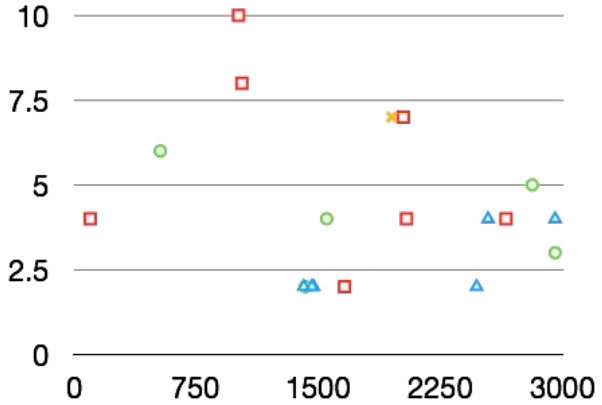
Previously Rutgers, Princeton (adjunct), AT&T, Duke, Bellcore.

PhD from here.

Machine learning (since 1990), sequential decision making.

# Machine Learning Classes at Brown

https://cab.brown.edu/ (search for machine learning)




**Tool for solving hard engineering problems:**
CSCI 1470: Deep Learning
CSCI 1410: Artificial Intelligence
CSCI 1460: Computational Linguistics
CSCI 2470: Deep Learning
CSCI 2951K: Topics in Collaborative Robotics
ENGN 2540: Audio and Speech Processing

**Mathematical and algorithmic foundations:**
APMA 2811W: Convex Analysis and Minimization Algorithms
CSCI 0530: Directions: The Matrix in Computer Science
CSCI 1550: Probabilistic Methods in Computer Science
CSCI 2951M: Advanced Algorithms Seminar
CSCI 1420: Machine Learning

**Model for cognition:**
CLPS 1950: Deep Learning in Brains, Minds and Machines

**Tool for statistical analysis and understanding data:**
CSCI 0100: Data Fluency for All
DATA 1010: An Introduction to Topics in Probability, Statistics, and ML
DATA 2020: Probability, Statistics, and Machine Learning: Advanced Methods
PHP 2650: Statistical Learning and Big Data
DATA 1030: Introduction to Topics in Data and Computation Science
DATA 2040: Data and Computational Science
ECON 1660: Big Data

# Machine Learning Classes at Brown

https://cab.brown.edu/ (list created by expert)

**Tool for solving hard engineering problems:**
CSCI 1470/2470: Deep Learning
CSCI 1410: Artificial Intelligence
CSCI 1460: Computational Linguistics
CSCI 2951K: Topics in Collaborative Robotics
ENGN 2540: Audio and Speech Processing

**Mathematical and algorithmic foundations:**
APMA 1690: Computational Probability and Statistics
APMA 1740: Recent Applications in Probability and Statistics
CSCI 1550: Probabilistic Methods in Computer Science
CSCI 1420: Machine Learning

**Model for cognition:**
CLPS 1950: Deep Learning in Brains, Minds and Machines

**Tool for statistical analysis and understanding data:**
CSCI 0100: Data Fluency for All
DATA 1010: An Introduction to Topics in Probability, Statistics, and ML
DATA 2020: Probability, Statistics, and Machine Learning: Advanced Methods
PHP 2650: Statistical Learning and Big Data
DATA 1030: Introduction to Topics in Data and Computation Science
DATA 2040: Data and Computational Science
ECON 1660: Big Data

# Course Communications

Our course website:
http://cs.brown.edu/courses/csci1420/index.html

We will be doing questions and announcements via Piazza. Please visit the course website to sign up.

# Grading Breakdown

- 8 homeworks, each with written & programming components (53%)
  - HW0: 4%
  - HW1-7: 7% each
  - 4 late days, up to 3 on any assignment
- 2 midterms (40%)
  - 20% each
  - 1st: March 20th (in-class)
  - 2nd: May 8th or May 16th (you get to choose)
- iClickers/REEF (7%)
  - Register before next class! (Instructions on the website)
  - 5 lowest scores dropped. No exceptions!

# Organizing Principle

ML algorithm = representation + loss function + optimizer

$$\hat{\min}_{r \in R} L_S(r)$$

Note:

- Optimizer might not be perfect (computationally intractable).
- Loss/error function is with respect to data.

# Syllabus (Part 1)

1. Discrete Structures
    a. R: Decision stumps, L: Training set error, O: Exhaustive search. PAC basics.
    b. R: Decision trees, O: Greedy search / local search.
    c. Cross validation, overfitting, complexity penalty.
2. Discrete Probabilistic Structures
    a. R: Naive Bayes, L: Likelihood, O: Maximum likelihood.
    b. Bayesian decision theory.
    c. R: Latent variable model, O: Expectation maximization.
    d. R: Bayesian non-parametric, O: MCMC.
3. Continuous Classification
    a. R: K nearest neighbors.
    b. R: K-means, mixture of Gaussians, vector quantization.
    c. R: Decision tree with ranges, rectangles. VC dimension, Rademacher.

# Syllabus (Part 2)

4. Continuous Classification
    a. R: Linear separators, O: Perceptron algorithm. L: Robustness, O: Linear programming.
    b. L: Margins, O: Gradient descent. O: SVM algorithm, L: Regularization.
    c. R: Kernel trick. Multiclass SVM.
    d. O: Boosting.
5. Continuous Regression
    a. R: Linear function, O: Least squares, R: Expanded feature space, O: MLE view. Outliers.
    b. PCA. O: Spectral methods.
    c. R: Feedforward neural networks, O: Gradient descent
    d. O: Random projections
6. Complex Inputs and Outputs
    a. R: Hidden Markov Models
    b. Sequence to sequence mapping
    c. Contextual bandits

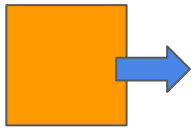
# Book

None required.

- Using http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/ . (pdf is downloadable.) Chapter 1 today. Chapter 2 and 18 next.

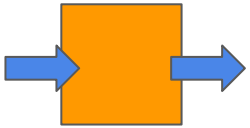
# Three Kinds of Programs

Generator



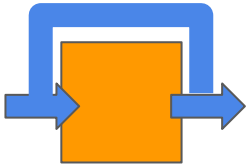
- Takes no input, produces output
- Compression (program represents output), algorithmic generation (random)

Function



- Maps input to output
- Transformations, query/response

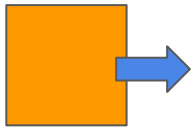
Interactive



- Takes some input, produces some output, expects more input
- Operating systems, games, UIs, robots

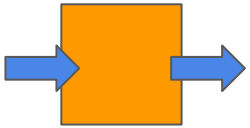
# Three Kinds of Machine Learning

Unsupervised



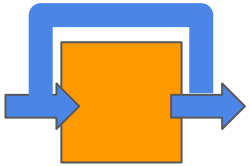
- Given data, finds a representation
- Descriptive: What happened?

Supervised



- Given input/output examples, finds mapping
- Predictive: What will happen?

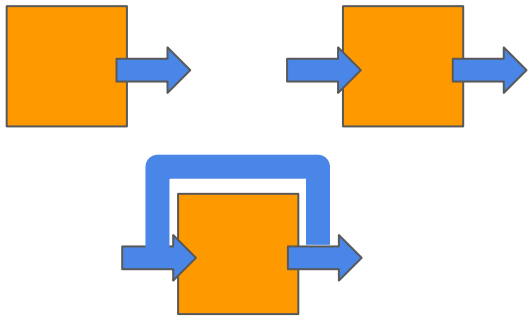
Reinforcement



- Translate state to action to maximize reward
- Prescriptive: What should we do?

# Machine Learning as Program Generation

Want to define a function.



**Representation**: Define a space of possible functions.

*Tradeoffs: Expressibility, simplicity, convenience...*

**Loss function**: Decide how to score a function.

*The data usually comes into play here.*

**Optimizer**: Search the space of functions for one with a high score.

*The best scoring function is the one returned.*

# Interactive Case Study

$k = 6$ students ($0$ through $k$-$1$) come to class each day. Arrival orderings:

(4, 0, 2, 3, 5, 1), (3, 5, 4, 0, 1, 2), (0, 3, 1, 5, 4, 2), (0, 2, 3, 1, 5, 4), (0, 4, 1, 3, 2, 5), (5, 4, 0, 2, 3, 1), (2, 3, 0, 4, 1, 5), (0, 5, 2, 1, 3, 4), (0, 2, 1, 4, 3, 5), (3, 5, 0, 1, 2, 4), (4, 2, 0, 3, 1, 5), (3, 4, 2, 0, 1, 5), (5, 0, 4, 3, 2, 1), (5, 0, 3, 1, 4, 2), (2, 5, 1, 3, 0, 4), (0, 4, 1, 3, 5, 2), (3, 5, 2, 4, 0, 1), (5, 2, 4, 0, 3, 1), (3, 5, 4, 2, 0, 1), (5, 4, 0, 3, 1, 2), (3, 5, 2, 4, 0, 1), (0, 1, 2, 5, 4, 3), (1, 0, 3, 4, 2, 5), (0, 2, 4, 5, 1, 3), (2, 3, 5, 4, 1, 0), (0, 1, 3, 2, 5, 4), (3, 4, 2, 0, 1, 5), (0, 2, 4, 3, 5, 1), (2, 0, 3, 4, 1, 5), (3, 2, 5, 1, 0, 4), (4, 5, 1, 2, 3, 0), (2, 0, 4, 3, 5, 1), (1, 2, 0, 4, 5, 3), (2, 4, 0, 1, 5, 3), (0, 5, 1, 4, 3, 2), (5, 1, 2, 4, 3, 0), (4, 2, 5, 3, 0, 1), (5, 3, 2, 0, 4, 1), (0, 1, 2, 3, 4, 5), …

Predict the next arrival ordering.

# Interactive Case Study

$k = 6$ students ($0$ through $k$-$1$) come to class each day. Arrival orderings:

(2, 3, 4, 5, 1, 0), (3, 5, 4, 1, 0, 2), (4, 2, 3, 0, 1, 5), (1, 4, 3, 5, 0, 2), (4, 3, 2, 5, 1, 0), (4, 1, 5, 2, 3, 0), (4, 0, 5, 1, 2, 3), (1, 4, 0, 3, 2, 5), (1, 5, 0, 4, 2, 3), (1, 0, 2, 3, 5, 4), (0, 5, 2, 3, 4, 1), (3, 1, 2, 5, 4, 0), (4, 2, 1, 0, 5, 3), (2, 5, 4, 3, 1, 0), (0, 5, 2, 4, 1, 3), (2, 3, 4, 5, 0, 1), (4, 1, 3, 0, 2, 5), (0, 4, 2, 1, 5, 3), (4, 1, 5, 0, 3, 2), (1, 2, 0, 3, 5, 4), (2, 5, 3, 1, 0, 4), (5, 2, 4, 3, 1, 0), (2, 0, 1, 5, 4, 3), (2, 5, 0, 4, 1, 3), (3, 5, 2, 0, 4, 1), (3, 0, 4, 2, 1, 5), (2, 0, 1, 5, 4, 3), (3, 2, 5, 4, 0, 1), (4, 0, 2, 3, 5, 1), (3, 0, 4, 1, 5, 2), (2, 0, 3, 4, 5, 1), (1, 0, 2, 4, 5, 3), (2, 0, 5, 1, 3, 4), (5, 1, 0, 4, 2, 3), (3, 4, 5, 1, 2, 0), (0, 2, 4, 5, 3, 1), (2, 1, 3, 0, 5, 4), (1, 3, 5, 4, 0, 2), (4, 1, 3, 5, 2, 0), … [5000 total]

Predict the next arrival ordering.

Person 0: 0.1

Person 1: 0.2

Person 2: 0.2

Person 3: 0.3

Person 4: 0.2

Ordering: 14203

# Approach 1: Voting

Count the frequencies of all of the permutations in the data.

Return the most common one.

Makes sense if we assume data is iid (independent and identically distributed).

Estimates $k!$ parameters.

(2, 0, 3, 4, 5, 1), 21)
(4, 3, 0, 2, 5, 1), 20)
(2, 3, 0, 4, 5, 1), 20)
(4, 5, 0, 3, 2, 1), 20)
(0, 2, 4, 5, 3, 1), 19)
(2, 0, 5, 3, 4, 1), 19)
(5, 2, 3, 0, 4, 1), 19)
(5, 0, 2, 3, 4, 1), 19)
(3, 2, 0, 5, 4, 1), 18)
(5, 2, 0, 3, 4, 1), 18)

LOSS:
**0.997260322933**

# Approach 2: The Matrix

Count the frequencies of each student in each position.

Return the most likely one.

Makes sense if we assume data is generated by flipping a coin for each student-position pair and then outputting it only if it forms a permutation.

Estimates $k^2$ parameters.

```
[[ 1019.  567. 877. 823.   889.  825.]
 [ 1015.  583. 937. 830.   836.  799.]
 [  888.  675. 862. 874.   882.  819.]
 [  816.  758. 847. 842.   855.  882.]
 [  686.  972. 806. 843.   850.  843.]
 [  576. 1445. 671. 788.   688.  832.]]
```

Amazingly, this optimization problem is efficiently solvable using the so-called “Hungarian Algorithm” for assignments.

| | LOSS: |
|---|---|
| (0, 2, 4, 3, 5, 1) | **0.997100677686** |

# Approach 3: Probability Estimation

Count the frequencies of each student in the first position.

Return the students in reverse order.

Makes sense if we assume data is generated by flipping a coin for each student-position pair and then outputting it only if it forms a permutation.

Estimates $k$ parameters.

Data was generated this way! Why worse??

(0, 1019)
(4, 889)
(2, 877)
(5, 825)
(3, 823)
(1, 567)

(0, 4, 2, 5, 3, 1)

LOSS:
**0.997189564172**

# Summary

| **Approach** | **Representation** | **Loss** | **Optimizer** |
|---|---|---|---|
| Voting | Perms ($\Pr(p)$) | $1-\Pr(p)$ | Max in list |
| The Matrix | Perms ($\Pr(s\|i)$) | $\prod_{(s,i)\in p} \Pr(s\|i)$ | Hungarian algorithm |
| Probability Estimation | Perms ($\Pr(s)$) | $\prod_s \Pr(s) \,/\, \prod_{s'<s} \Pr(s')$ | Sort student probs |

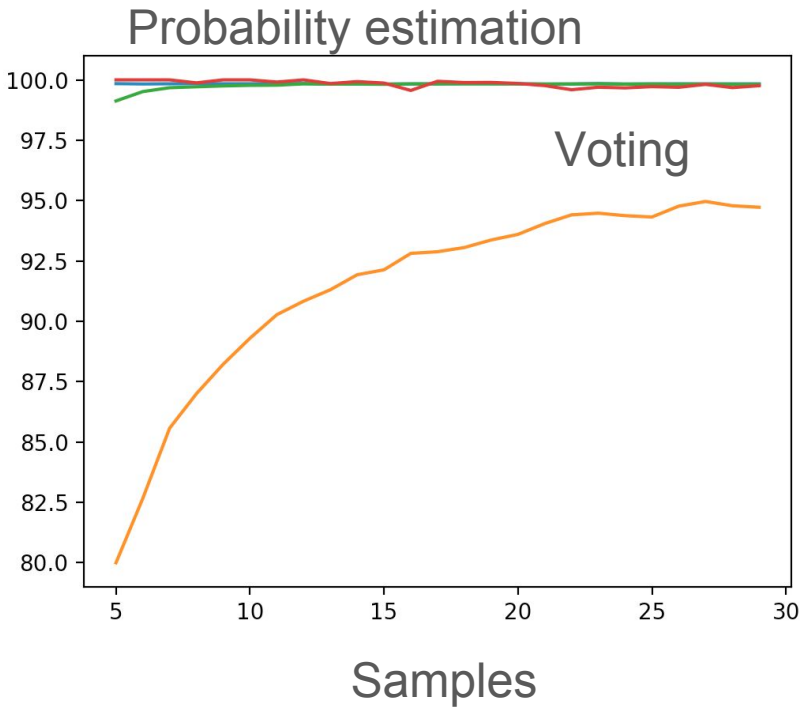
Key elements:

- Does loss on the data relate to the actual problem to be solved?
- ...with a tractable amount of data?
- Is optimization efficient?
- ...if approximate, do we lose our guarantee above?

# Notes on Case Study

Overfitting.

What's the true metric we care about? What's the space of representations? How do we compute a loss for a possible representation? How do we search for one that minimizes loss? How do we relate this loss to the actual metric we care about? Can we do better?



# Some Historical Highlights

Rosenblatt, perceptrons
Samuelson, checker player
Minsky and Papert, perceptrons
Vapnik and Chervoninkis, VC
Hinton, Boltzmann machine / xor
HMMs for speech
Sejnowski, Nettalk
Valiant, PAC
Pomerleau, Navlab
Lecun, Check reader
Tesauro, Backgammon

Ng, helicopter tricks
Netflix prize
Bandit algorithms for web ads
Face detection in cameras
IBM, Jeopardy
Google, unsupervised images (cats)
Imagenet
Speech recognition
Deepmind, Atari learner
Deepmind, Alpha*
Image manipulation goes viral

# AMA

What’s ML about?

# Overfitting is Scary

https://www.youtube.com/watch?v=DQWI1kvmwRg