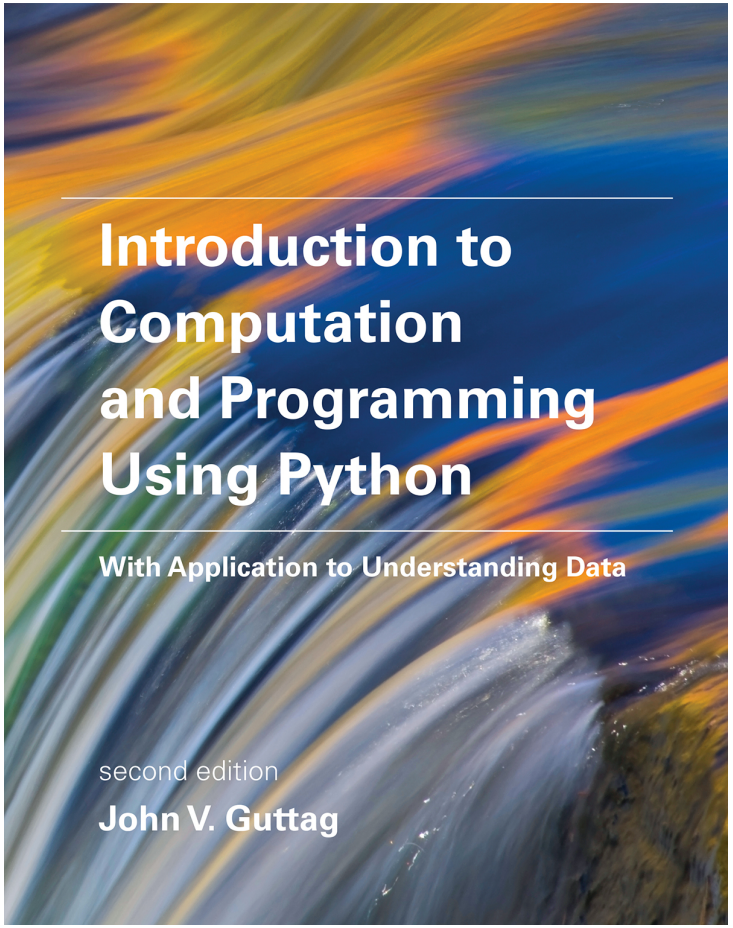
Introduction to Machine Learning

John Guttag

MIT Department of Electrical Engineering and Computer Science

Assigned Reading

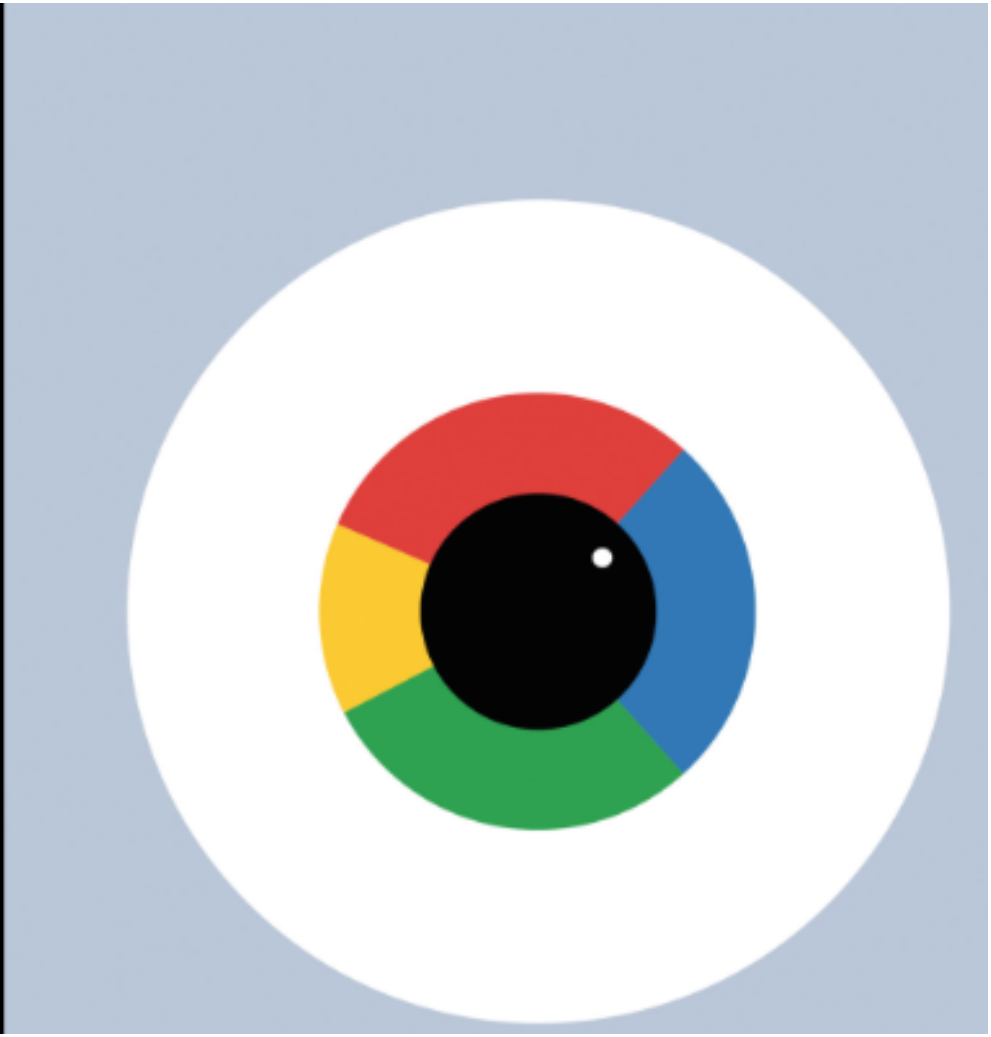
- Today:
  - Chapter 22
- Next lecture:
  - Chapter 23



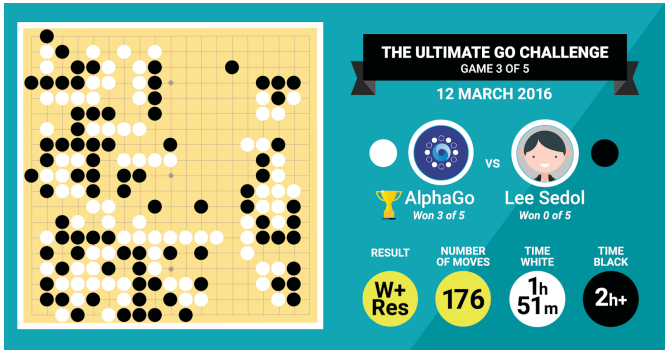
The Plan Ahead

- Machine learning is a huge topic
  - Courses covering machine learning include 6.008, 6.036, 6.860, 6.860, 6.862, 6.867
  - Topic is large component of other courses, e.g., in natural language processing, computational biology, computer vision, robotics, other areas
- In 6.0002, we will
  - Provide an introduction to the basic ideas:
  - Introduce classification methods (learning with outcomes), such as “k nearest neighbor” methods
  - Introduce clustering methods (learning without outcomes), such as “k-means”

Why Talk About Machine Learning?



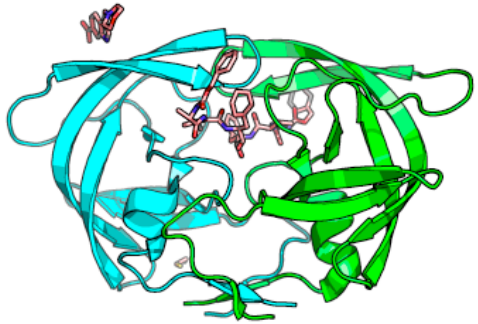
Machine Learning Is Everywhere



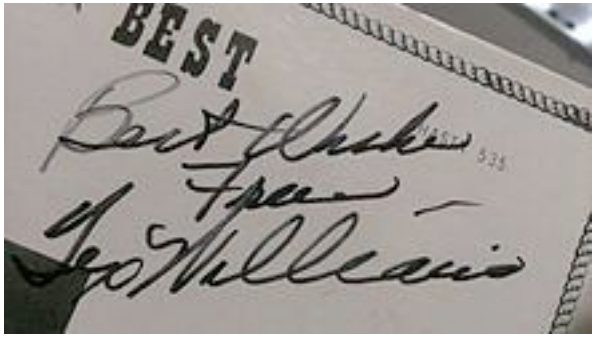
AlphaGo

NETFLIX

Recommendation systems



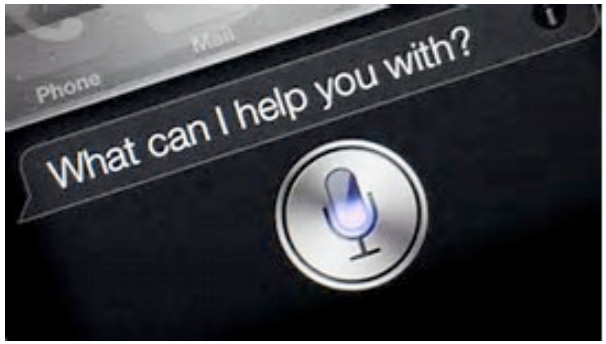
Drug discovery



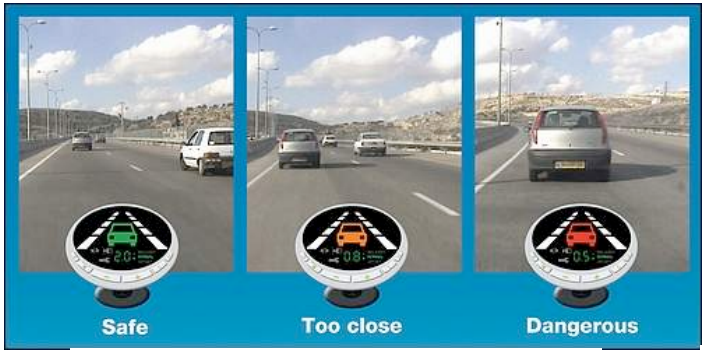
Character recognition

2σ TWO SIGMA

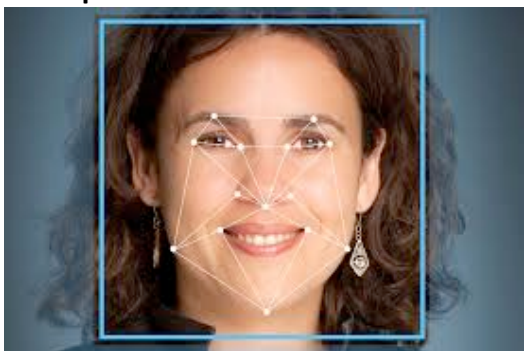
Hedge fund stock predictions



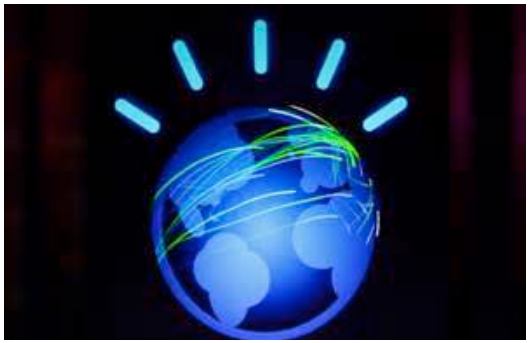
Voice assistants



Assisted driving



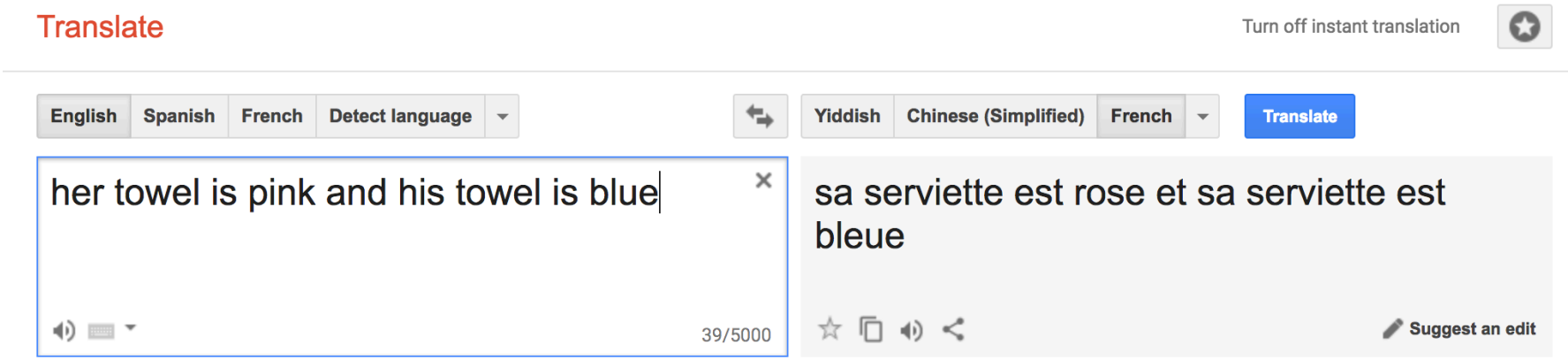
Face detection/recognition



Cancer diagnosis

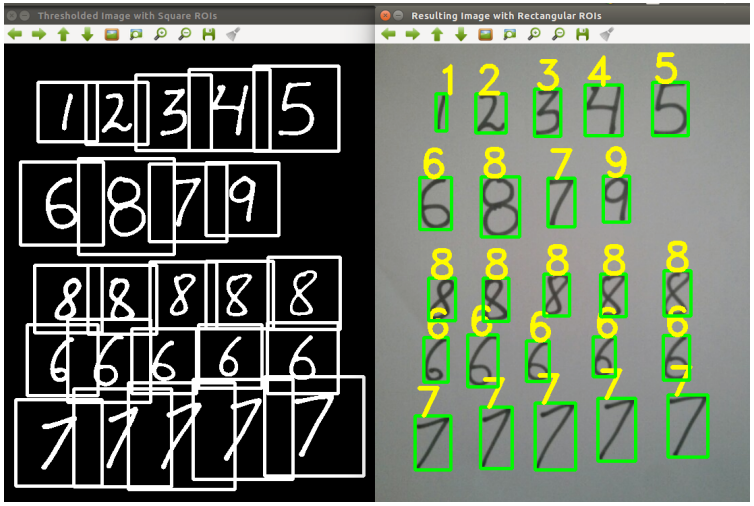
Success stories: Speech & Language

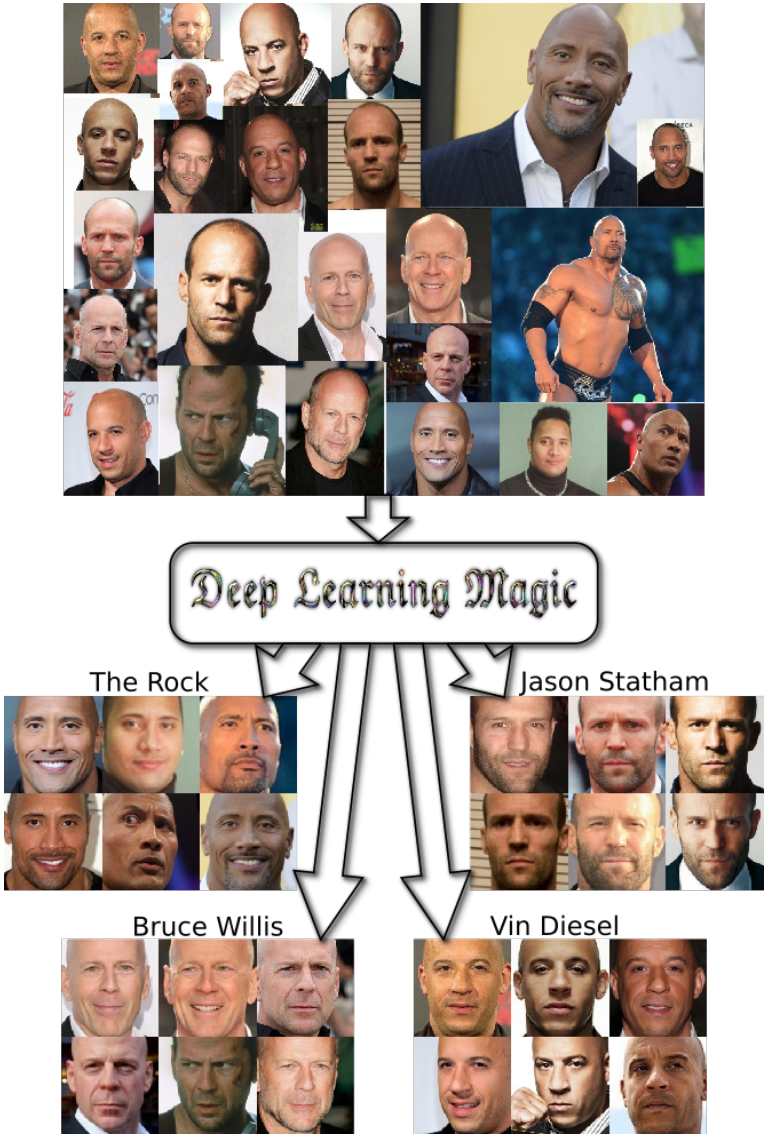
- Many applications already available
  - Apple Siri
  - Amazon Echo
  - Google Translate
  - Baidu Deep Voice



Success stories: Vision

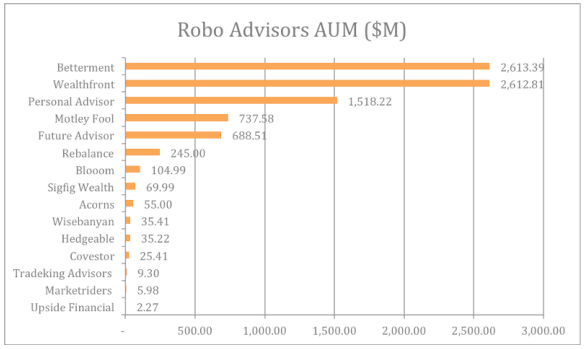
- Face recognition
- Postal service uses handwriting recognition





Success stories: Finance

- Nine out of top ten hedge funds last year used algorithmic trading
  - While kept secret, most (e.g., Two Sigma) extensively use machine learning tools
- Web fraud detection based on behavioral patterns
- Predicting consumer credit risk for loan underwriting
- Robo-advisors for portfolio management



Success stories: Game players

- Google DeepMind
  - Uses a Monte Carlo tree search algorithm to find moves, based on knowledge learned using an artificial neural network (ANN) trained against humans and itself
  - Uses reinforcement learning on an ANN to refine model
  - Method initially learned from examples selected by human; has been applied to learning without explicit knowledge – given an objective function, it uses reinforcement to learn good strategies

Playing preview

What Is Machine Learning?

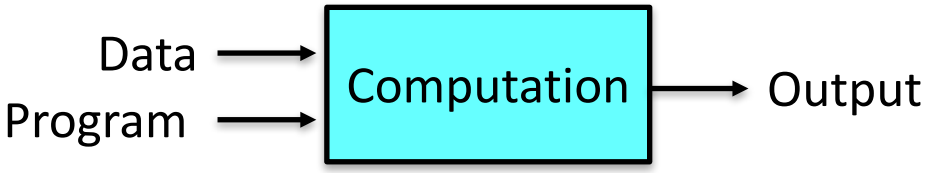
- All useful programs "learn" something
- In the first lecture of 6.0001 we looked at an algorithm for finding (learning?) square roots
- We recently looked at using linear regression to find (learn?) a model of a collection of points
- We could argue that root finding and curve fitting algorithms "learn" models to fit to data sets
- But each algorithm is designed to meet a specific goal, and somehow machine learning should be broader than that

What Is Machine Learning?

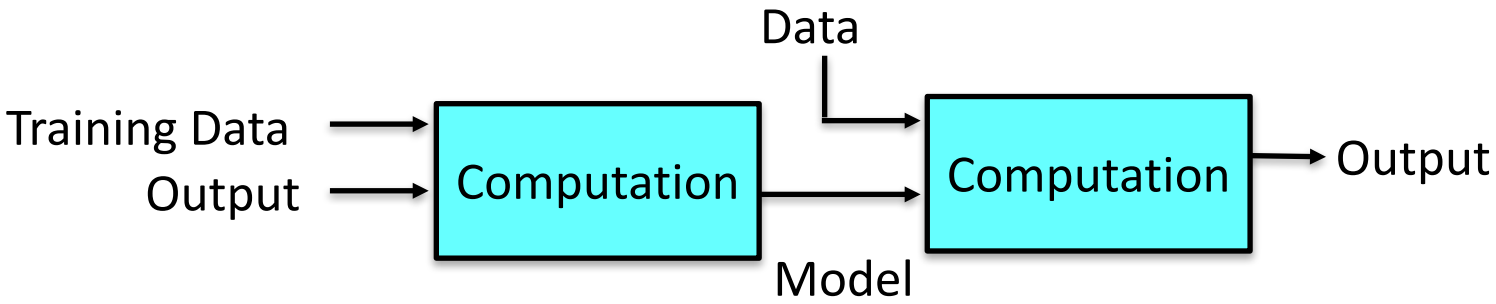
- Early definition of machine learning:
  - *"Field of study that gives computers the ability to learn without being explicitly programmed."* Arthur Samuel (1959)
    - Computer pioneer who wrote first self-learning program, which played checkers – learned from "experience"
    - Invented alpha-beta pruning – widely used in decision tree searching
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."* Tom Mitchell – CMU (1997)

What Is Machine Learning

Traditional Programming



(Supervised) Machine Learning



AlphaGo was learned to play Go by learning a model from training data selected by people (the training sets were the "programming")

AlphaGo Zero learned to play Go with no human input, just just by playing against itself, based on an objective function and a set of rules

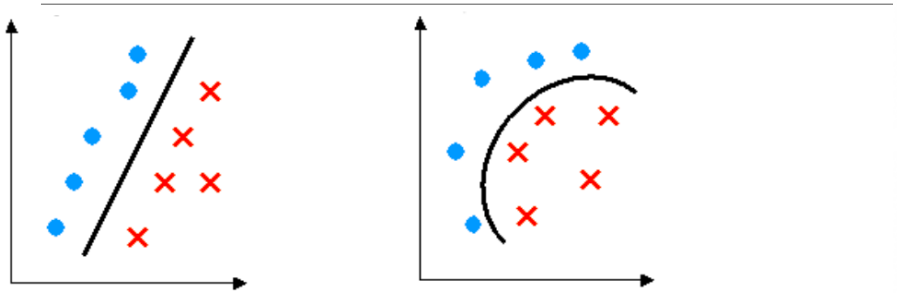
How Are Things Learned?

- Memorization
  - Accumulation of individual facts
  - Limited by
    - Time to observe facts
    - Memory to store facts
- Generalization
  - Deduce new facts from old facts
  - Limited by accuracy of deduction process
    - Essentially a predictive activity
    - Assumes that the past predicts the future
- Extend deduction to programs that can infer useful information from **implicit** patterns in data

Basic Paradigm

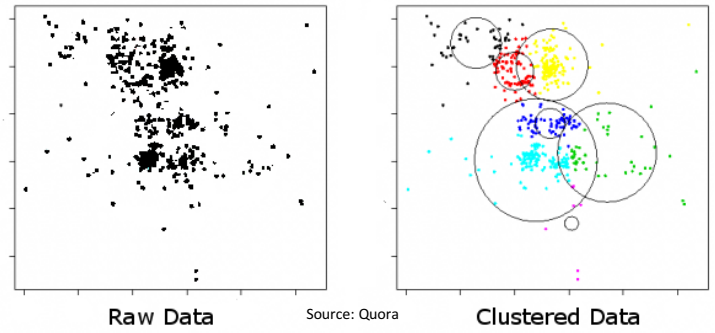
- Observe set of examples: training data
- Infer something about process that generated that data – learn a model that predicts data
  - Regression: prediction is continuous
    - E.g., predict what a student's GPA will
  - Classification: prediction is categorical
    - E.g., predict whether a student will major in CS
- Use inference to make predictions about previously unseen data: test data

Variations on Paradigm



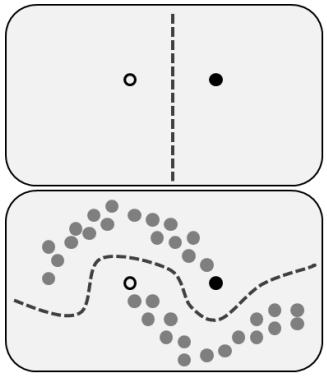
Source: Utah CS

Supervised Learning



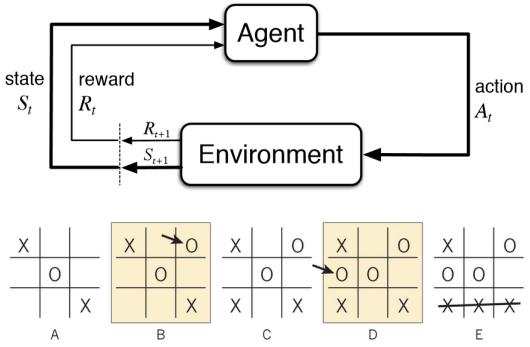
Source: Quora

Unsupervised Learning



Source: Wikipedia

Semi-Supervised Learning



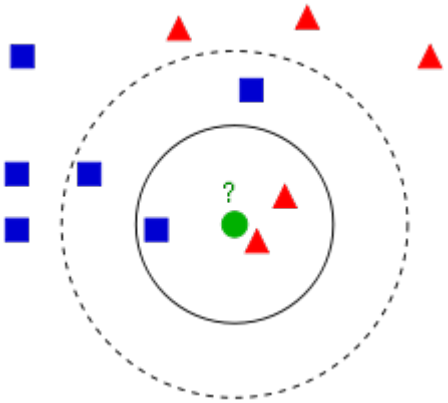
Source: Stanford CS and Nature

Reinforcement Learning

Unsupervised and Supervised Learning

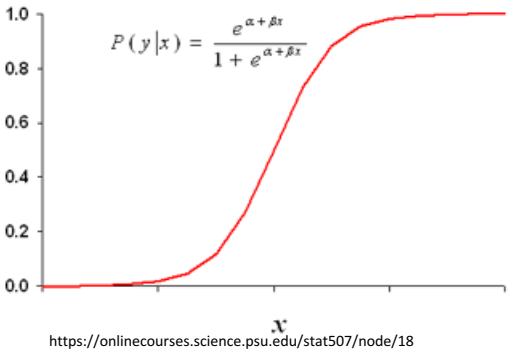
- Unsupervised learning
  - Given set of unlabeled examples
  - Convert each into a vector of features
  - Cluster based on similarity of feature vectors (e.g., k means)
- Supervised learning
  - Given set of labeled examples
  - Convert each into a vector of features
  - Use algorithm to learn coefficient of variables to optimize tradeoff of selectivity and specificity

Some Algorithms



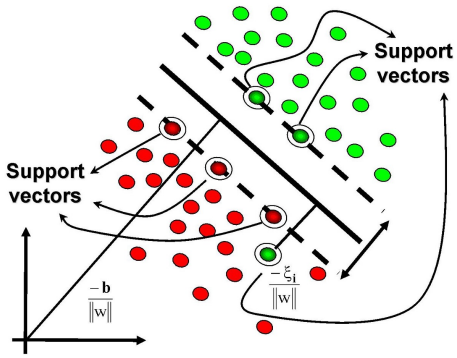
https://onlinecourses.science.psu.edu/stat507/node/18

KNN
1951 (Fix et al.)



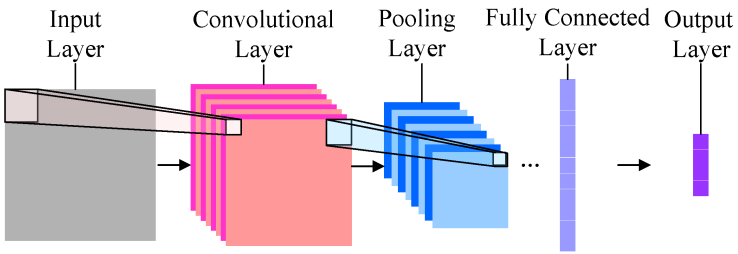
https://onlinecourses.science.psu.edu/stat507/node/18

Logistic Regression
1958 (Cox)



Source: https://medium.com/@haydar_ai/learning-data-science-day-11-support-vector-machine-8ef06da91bfc

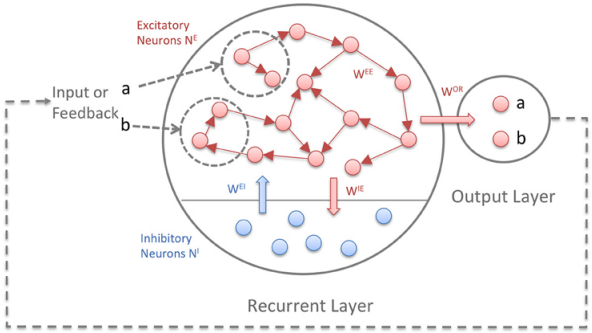
Support Vector Machines
1963/1992 (Vapnik et al.)



Source: http://www.mdpi.com

Convolutional Neural Networks

Neural Networks
1957/1986/1998/2006/2012



Source: www.frontiersin.org/articles/10.3389/fncom.2015.00036/full

Recurrent Neural Networks

All ML Methods Require:

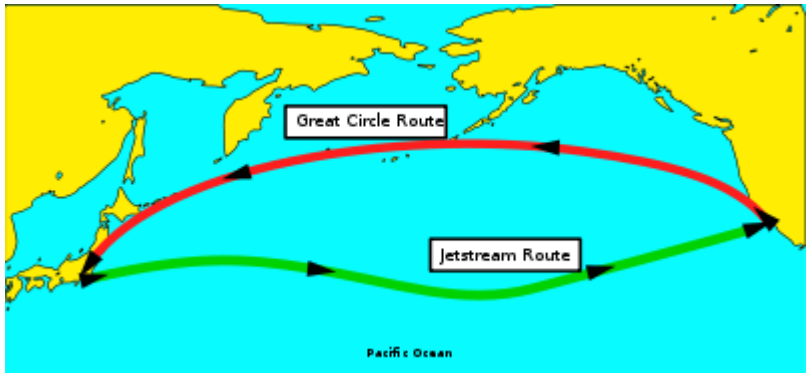
- Choosing training data and evaluation method
- Representation of the features
- Distance metric for feature vectors
- Objective function and constraints
- Optimization method for learning the model

Rest of Today

Next two lectures

Setting Up the Learning Framework

- How are we going to represent our training data?
  - What features are important?
  - How are they represented? (Typically we want features that can be mapped to numerical values, so we can measure distances between examples)
    - Binary
    - Integers
    - Floats
- How do we measure distances between feature vectors representing instances?
  - Relative scales of axes
  - Distance metric



Feature Representation

- Features never fully describe the situation
- Feature engineering
  - Represent examples by feature vectors that will facilitate generalization
    - Suppose I want to use 100 examples from past to predict, at the start of the subject, which students will get an A in 6.0002
    - Some features surely helpful, e.g., GPA, prior programming experience (not a perfect predictor), mathematical sophistication
    - Others might cause me to overfit, e.g., birth month, eye color
- Want to maximize ratio of useful input to irrelevant input in choice of features
  - Signal-to-Noise Ratio (SNR)

An Example

Features **Label**

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes

Initial model:

- Everything is a reptile (no features needed)

An Example

	Features					Label
Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Chicken	True	True	False	False	2	No

New model use feature:

- Poisonous or # legs

An Example

Features Label

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Chicken	True	True	False	False	2	No
Boa constrictor	False	True	False	True	0	Yes

Current model:

- Poisonous

Boa doesn't fit model, but is labeled as reptile.
Need to refine model

New Model:

- # legs



An Example

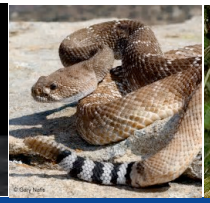
Features **Label**

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No

Current model:

- # legs

Still okay

An Example

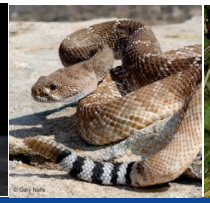
Features **Label**

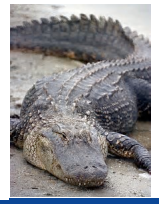
Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes

Current model:
- No legs

New model:
- Scales
- Cold blooded
- Legs != 2

Alligator doesn't fit model, but is labeled as reptile.
Need to refine model

An Example

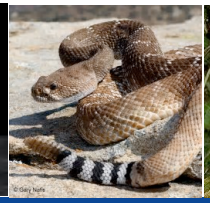
Features **Label**

Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes
Dart frog	True	False	True	False	4	No

Current model:

- Has scales
- Cold blooded
- Legs != 2

Still okay


An Example

	Features					Label
Name	Egg-laying	Scales	Poisonous	Cold-blooded	# legs	Reptile
Cobra	True	True	True	True	0	Yes
Rattlesnake	True	True	True	True	0	Yes
Boa constrictor	False	True	False	True	0	Yes
Chicken	True	True	False	False	2	No
Alligator	True	True	False	True	4	Yes
Dart frog	True	False	True	False	4	No
Salmon	True	True	False	True	0	No
Python	True	True	False	True	0	Yes

Time to find a way to generate classification rules

How do we learn to assign labels to examples?

- Have sets of examples represented as points in a feature space
- Intuition – similar examples or objects are close to one another
  - Do similar examples form one cluster in feature space, or several?
  - Which features are most important in grouping examples?
- Goal is to find way to group similar objects
  - Use distance between examples to determine important features and to identify new instances by type

Issues to consider in measuring distances

- **Feature engineering:**
  - Deciding which features to include and which are merely adding noise to classifier

You've seen this! – variant of overfitting

  - Defining how to measure distances between training examples (and ultimately between classifiers and new instances)
  - Deciding how to weight relative importance of different dimensions of feature vector, which impacts definition of distance

You've seen this! – also a variant of overfitting

Measuring Distance Between Animals

- We can think of our animal examples as consisting of four binary features and one integer feature
- One way to learn to separate reptiles from non-reptiles is to measure the distance between pairs of examples, and use that:
  - To cluster nearby examples into a common class (unlabeled data), or
  - To find a classifier surface in space of examples that optimally separates different (labeled) collections of examples from other collections

```
rattlesnake = [1,1,1,1,0]
boa constrictor = [0,1,0,1,0]
dart Frog = [1,0,1,0,4]
```

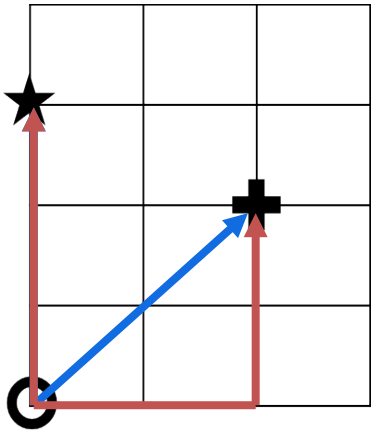
Can convert examples into feature vectors

Minkowski Metric

$$dist(X1, X2, p) = \left(\sum_{k=1}^{len} abs(X1_k - X2_k)^p\right)^{1/p}$$

p = 1: Manhattan Distance
p = 2: Euclidean Distance

Need to measure distances between feature vectors



Typically use Euclidean metric; Manhattan may be appropriate if different dimensions are not comparable

Is circle closer to star or cross?

- Euclidean distance
  - Cross – 2.8
  - Star – 3
- Manhattan Distance
  - Cross – 4
  - Star - 3

Euclidean Distance Between Animals

```
rattlesnake = [1,1,1,1,0]
boa constrictor = [0,1,0,1,0]
dartFrog = [1,0,1,0,4]
```

Euclidean Distance Between Animals

```
rattlesnake = [1,1,1,1,0]
boa constrictor = [0,1,0,1,0]
dartFrog = [1,0,1,0,4]
```

	rattlesnake	boa constrictor	dart frog
rattlesnake	--	1.414	4.243
boa constrictor	1.414	--	4.472
dart frog	4.243	4.472	--

Using Euclidean distance, rattlesnake and boa constrictor are much closer to each other, than either is to dart frog

Add an Alligator

- `alligator = Animal('alligator', [1,1,0,1,4])`
- `animals.append(alligator)`
- `compareAnimals(animals, 3)`

Add an Alligator

```
alligator = Animal('alligator', [1,1,0,1,4])
animals.append(alligator)
compareAnimals(animals, 3)
```

	rattlesnake	boa constrictor	dart frog	alligator
rattlesnake	--	1.414	4.243	4.123
boa constrictor	1.414	--	4.472	4.123
dart frog	4.243	4.472	--	1.732
alligator	4.123	4.123	1.732	--

Alligator is closer to dart frog than to snakes – why?

- Alligator differs from frog in 3 features, from boa in only 2 features
- But scale on “legs” is from 0 to 4, on other features is 0 to 1
- “legs” dimension is disproportionately influential

Using Binary Features

```
rattlesnake = [1,1,1,1,0]
boa constrictor = [0,1,0,1,0]
dartFrog = [1,0,1,0,1]
Alligator = [1,1,0,1,1]
```

	rattlesnake	boa constrictor	dart frog	alligator
rattlesnake	--	1.414	1.732	1.414
boa constrictor	1.414	--	2.236	1.414
dart frog	1.732	2.236	--	1.732
alligator	1.414	1.414	1.732	--

Now alligator is closer to snakes than it is to dart frog – makes more sense

Feature Engineering Matters

Binarization vs. Scaling

- Suppose we care about number of legs, not just whether animal has legs
- Scaling a more general solution
  - Scale each feature separately

```
def scaleFeature(vals):
    vals = np.array(vals)
    mean = sum(vals)/len(vals)
    sd = np.std(vals)
    vals = vals - mean
    return vals/sd
```

Z-Scaling

Mean = ?

Std = ?

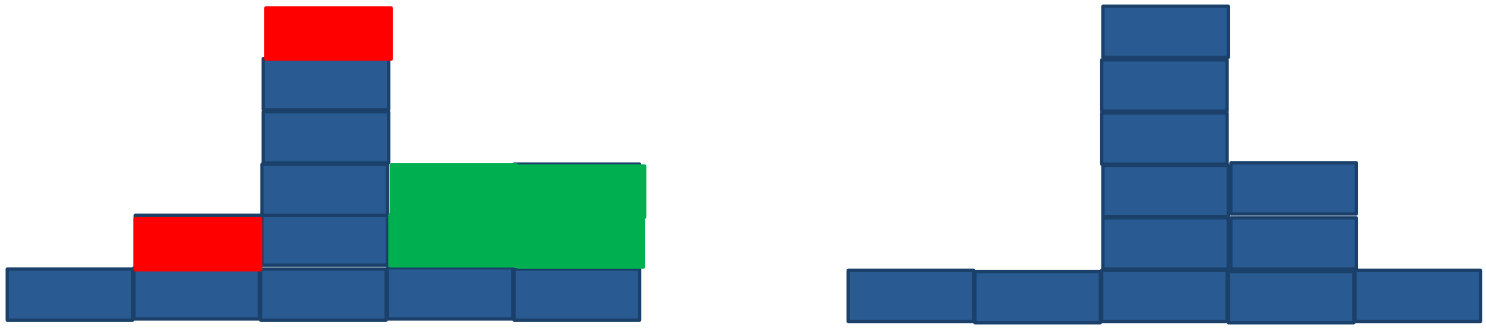
Other distance metrics

- Minkowski distance is commonly used because it naturally supports gradient descent methods
- But there are other distance metrics that can apply in comparing two examples to decide similarity
- One common metric:
  - Earth mover's distance (EMD)

Earth Mover's Distance

- Given two probability distributions (or histograms), what is the minimum amount of matter (dirt) that has to be moved (cost is amount to move times distance moved) to make the distributions match

Example of Earth Mover's distance



- Cost is 1 mass unit by 1 distance unit – 1
- Cost is 2 mass units by 1 distance unit – 2
- Total cost is 3 mass-distance units

Some observations

- Ordering of bins on axis is important, since distance to move “dirt” depends on this
- Makes sense to use EMD when
  - Applying to probability distributions or to other histograms with an inherent order to bins
    - Intensities in an image
    - Colors in an image
  - Or applying to settings with inherent spatial ordering
    - Object movement in frames of a video sequence
    - Words in a text document

Coming Up

- In the next two lectures, we will see examples of learning algorithms:
- When given unlabeled data, try to find clusters of examples near each other
  - Use centroids of clusters as definition of each learned class
  - New data assigned to closest cluster
- When given labeled data, learn mathematical surface that "best" separates labeled examples
  - New data assigned to class based on portion of feature space carved out by classifier surface in which it lies

But First