CIS 419/519

Introduction to Machine Learning

Instructor: Eric Eaton

www.seas.upenn.edu/~cis519

What is Machine Learning?

"Learning is any process by which a system improves performance from experience."

- Herbert Simon

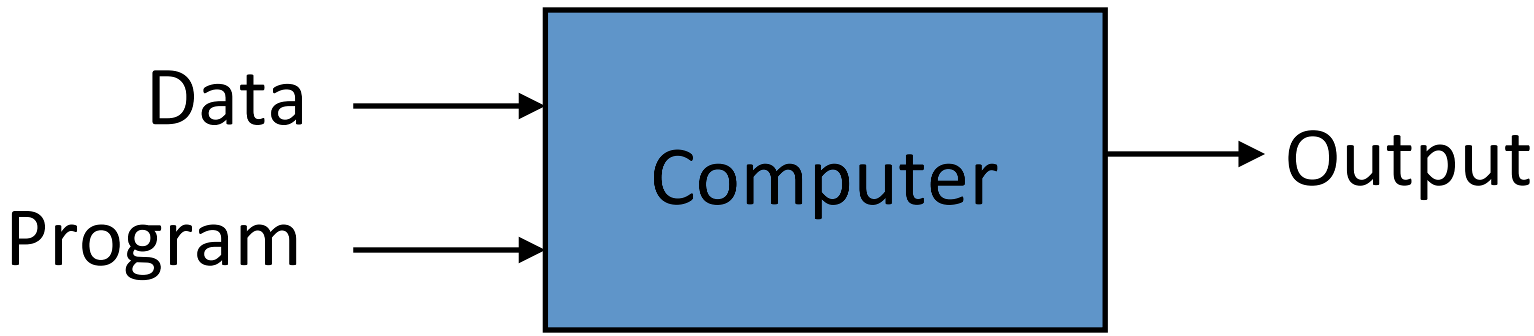
Definition by Tom Mitchell (1998):

Machine Learning is the study of algorithms that

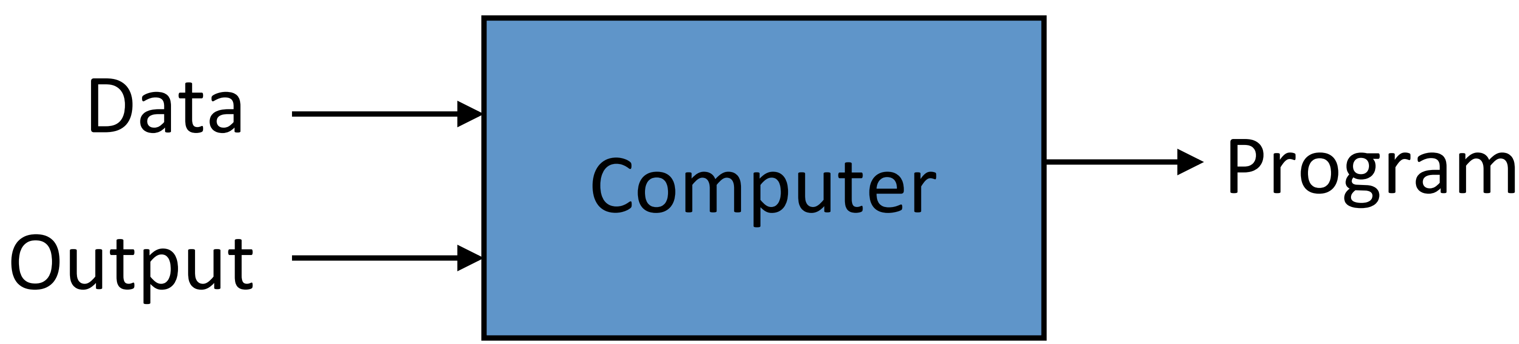
- improve their performance P
- at some task T
- with experience E.

A well-defined learning task is given by <P, T, E>.

Traditional Programming



Machine Learning



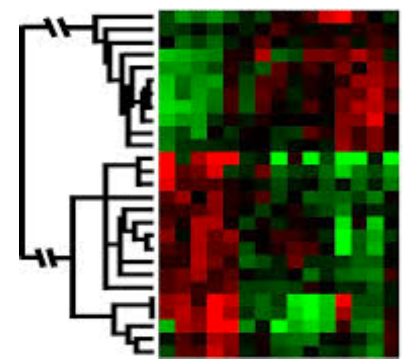
Slide credit: Pedro Domingos

When Do We Use Machine Learning?

ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)

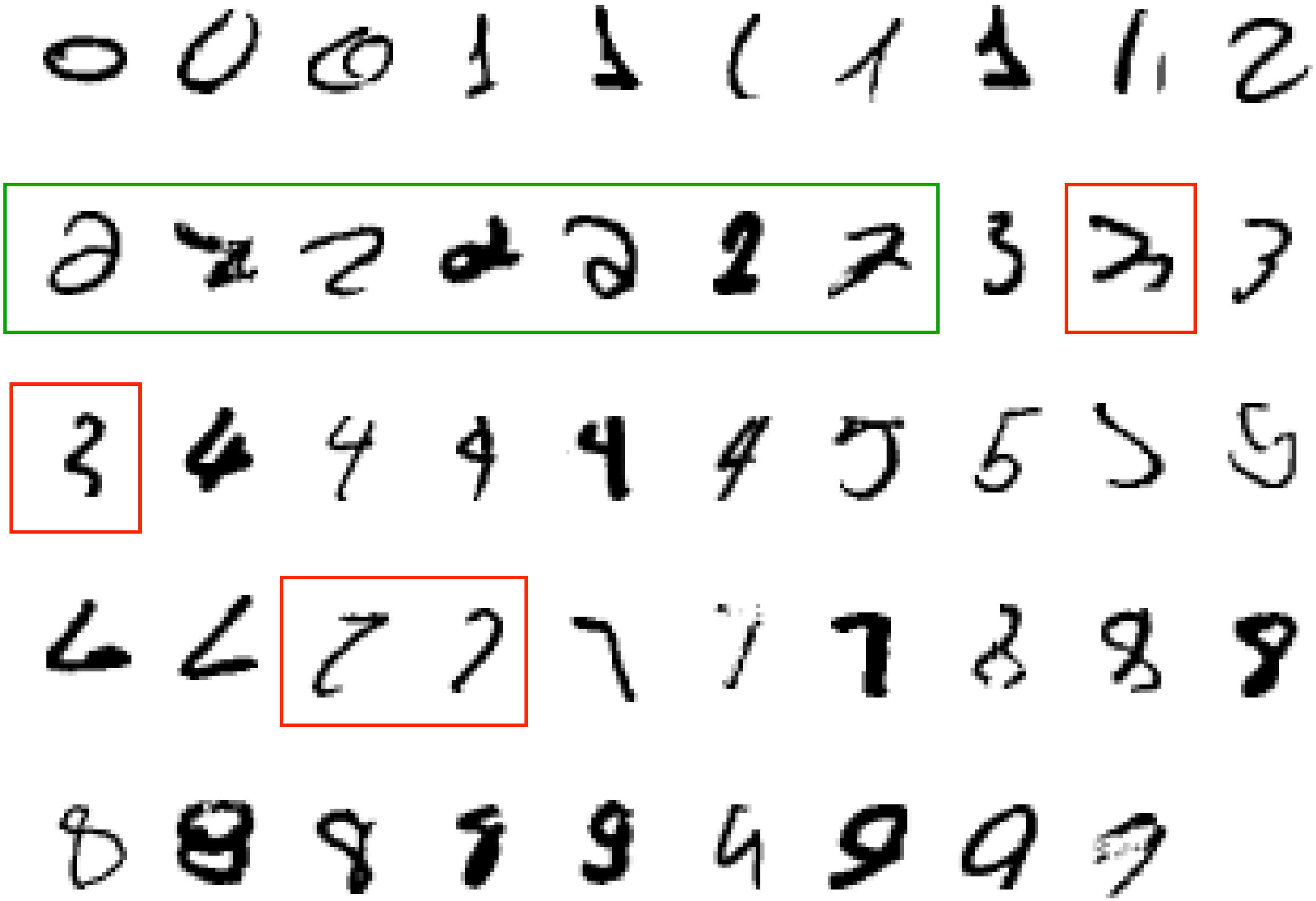


Learning isn't always useful:

- There is no need to "learn" to calculate payroll

A classic example of a task that requires machine learning: It is very hard to say what makes a 2



Slide credit: Geoffrey Hinton

Some more examples of tasks that are best solved by using a learning algorithm

- Recognizing patterns:
  - Facial identities or facial expressions
  - Handwritten or spoken words
  - Medical images
- Generating patterns:
  - Generating images or motion sequences
- Recognizing anomalies:
  - Unusual credit card transactions
  - Unusual patterns of sensor readings in a nuclear power plant
- Prediction:
  - Future stock prices or currency exchange rates

Slide credit: Geoffrey Hinton

Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging software
- [Your favorite area]

Slide credit: Pedro Domingos

Samuel's Checkers-Player

"Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed."   -Arthur Samuel (1959)

Defining the Learning Task

Improve on task T, with respect to performance metric P, based on experience E

T: Playing checkers
P: Percentage of games won against an arbitrary opponent
E: Playing practice games against itself

T: Recognizing hand-written words
P: Percentage of words correctly classified
E: Database of human-labeled images of handwritten words

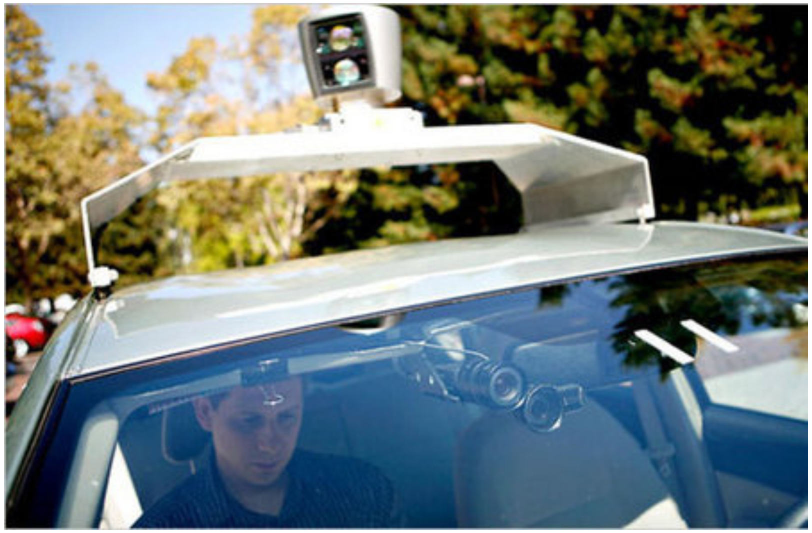
T: Driving on four-lane highways using vision sensors
P: Average distance traveled before a human-judged error
E: A sequence of images and steering commands recorded while observing a human driver.

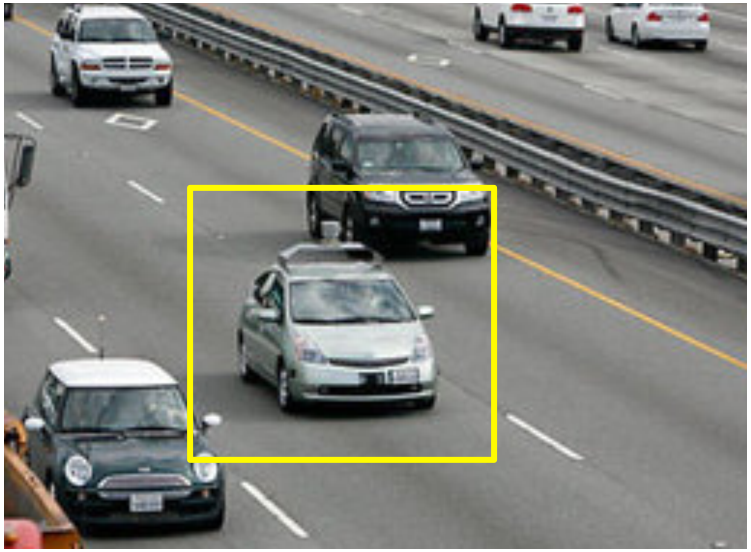
T: Categorize email messages as spam or legitimate.
P: Percentage of email messages correctly classified.
E: Database of emails, some with human-given labels

Slide credit: Ray Mooney

State of the Art Applications of Machine Learning

Autonomous Cars

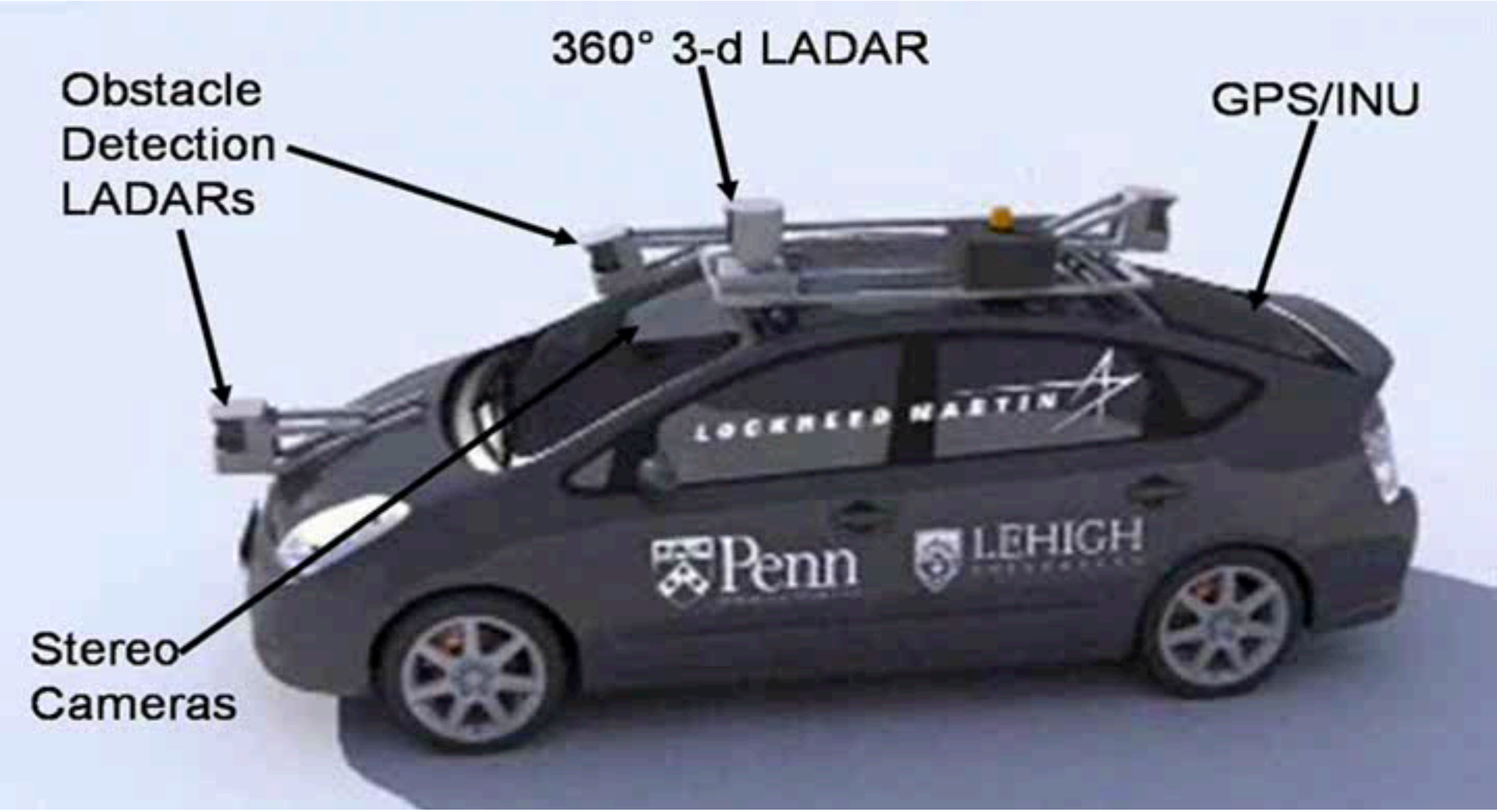




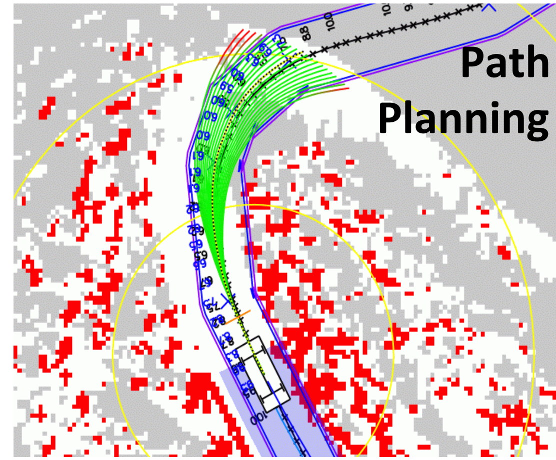
- Nevada made it legal for autonomous cars to drive on roads in June 2011
- As of 2013, four states (Nevada, Florida, California, and Michigan) have legalized autonomous cars

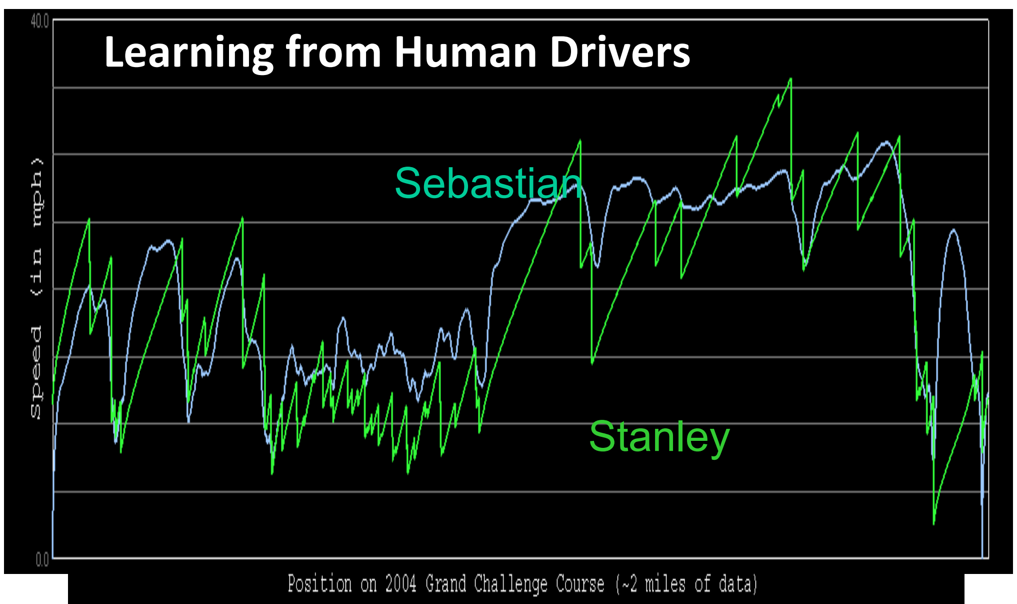
Penn's Autonomous Car →
(Ben Franklin Racing Team)

Autonomous Car Sensors



Autonomous Car Technology





Images and movies taken from Sebastian Thrun's multimedia website.

Deep Learning in the Headlines

BUSINESS NEWS

MIT Technology Review

Is Google Cornering the Market on Deep Learning?

A cutting-edge corner of science is being wooed by Silicon Valley, to the dismay of some academics.

By Antonio Regalado on January 29, 2014

How much are a dozen deep-learning researchers worth? Apparently, more than $400 million.

This week, Google reportedly paid that much to acquire DeepMind Technologies, a startup based in

Bloomberg Businessweek Technology

Acquisitions

The Race to Buy the Human Brains Behind Deep Learning Machines

By Ashlee Vance | January 27, 2014

intelligence projects. “DeepMind is bona fide in terms of its research capabilities and depth,” says Peter Lee, who heads Microsoft Research.

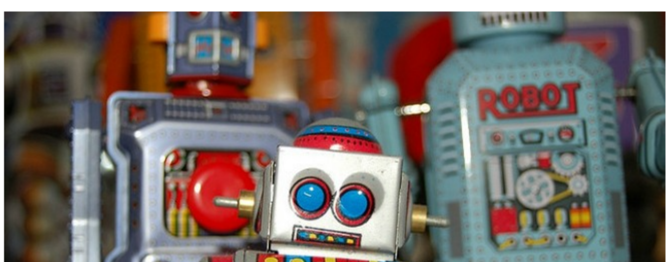
According to Lee, Microsoft, Facebook (FB), and Google find themselves in a battle for deep learning talent. Microsoft has gone from four full-time deep learning experts to 70 in the past three years. “We would have more if the talent was there to

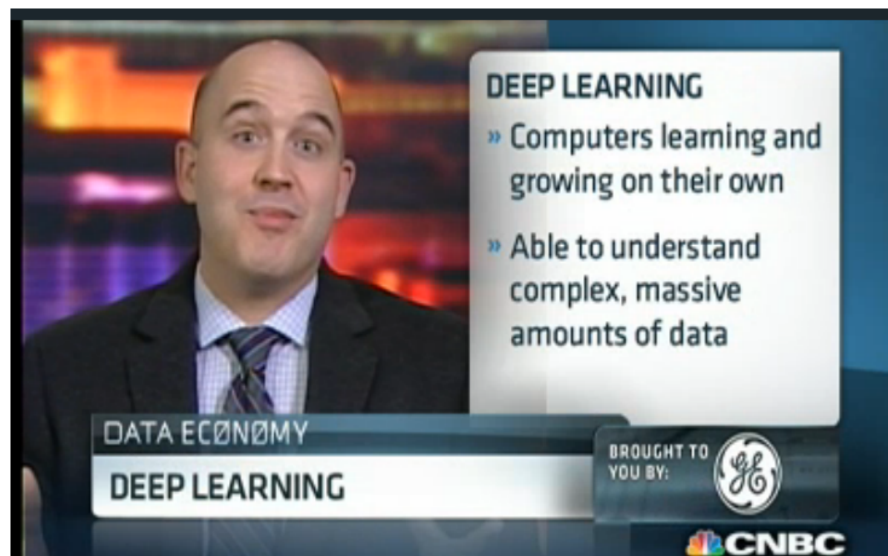
WIRED GEAR SCIENCE ENTERTAINMENT BUSINESS SECURITY DESIG

INNOVATION INSIGHTS community content featured

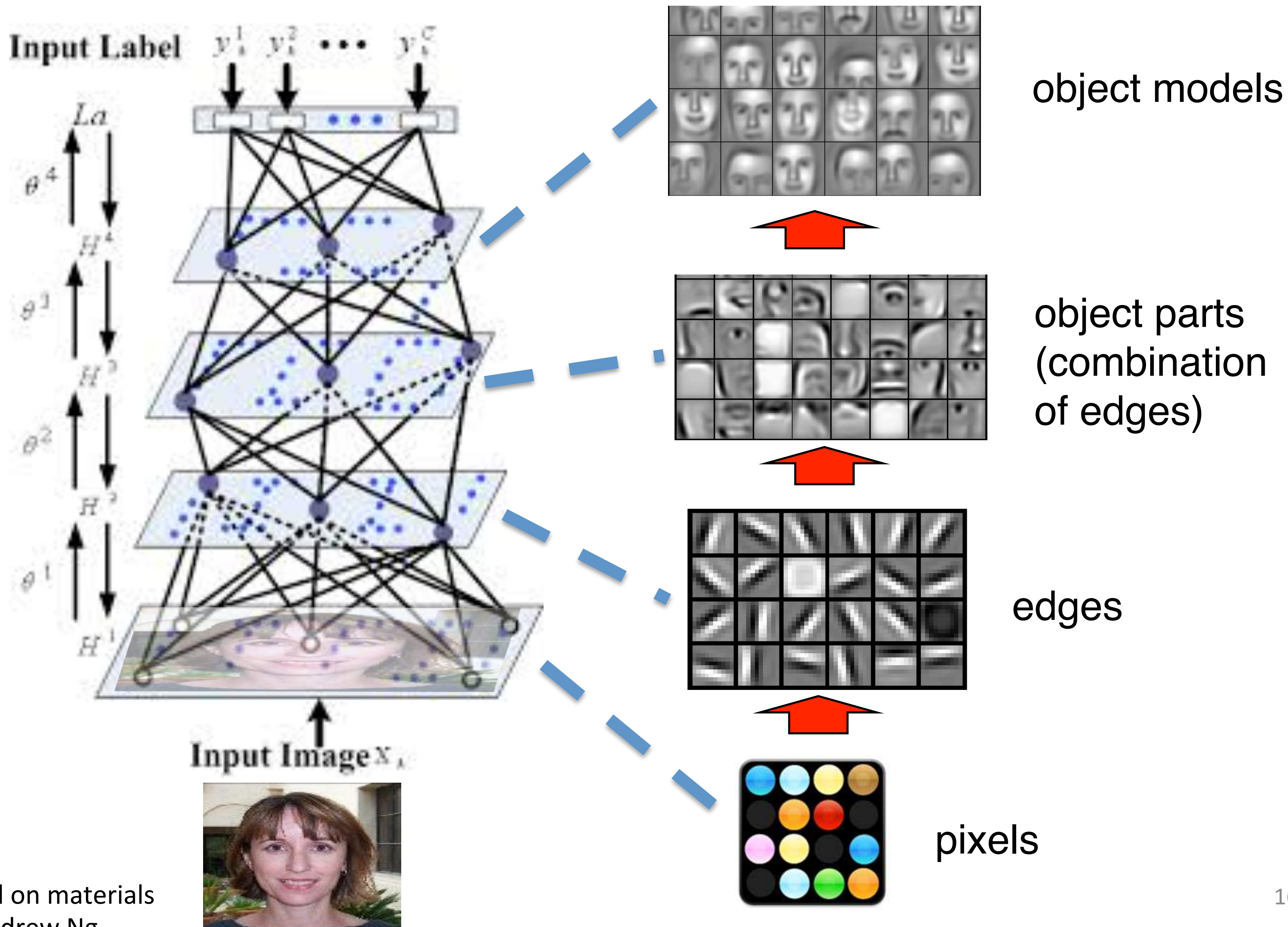
Deep Learning’s Role in the Age of Robots

BY JULIAN GREEN, JETPAC 05.02.14 2:56 PM



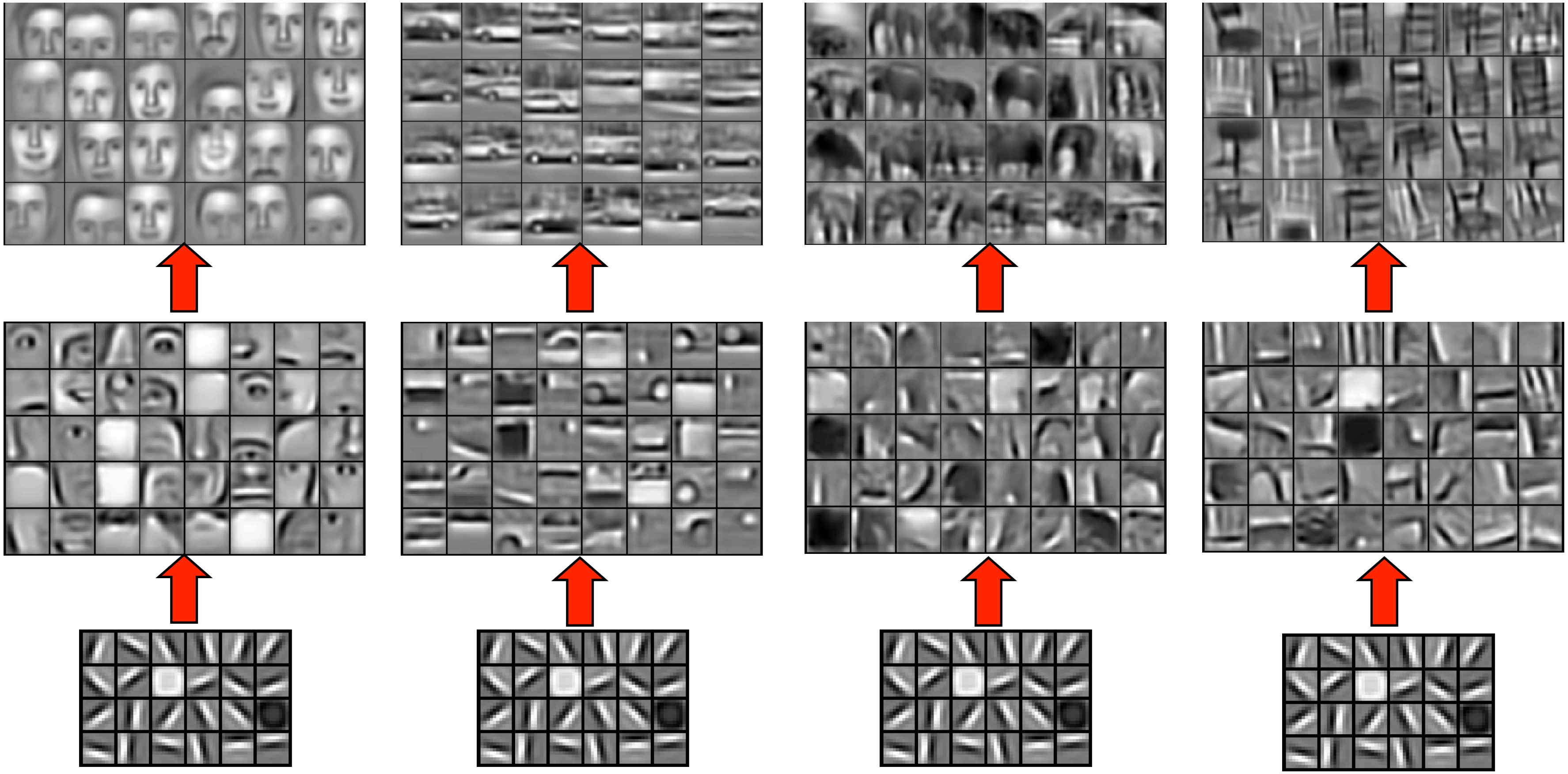


Deep Belief Net on Face Images



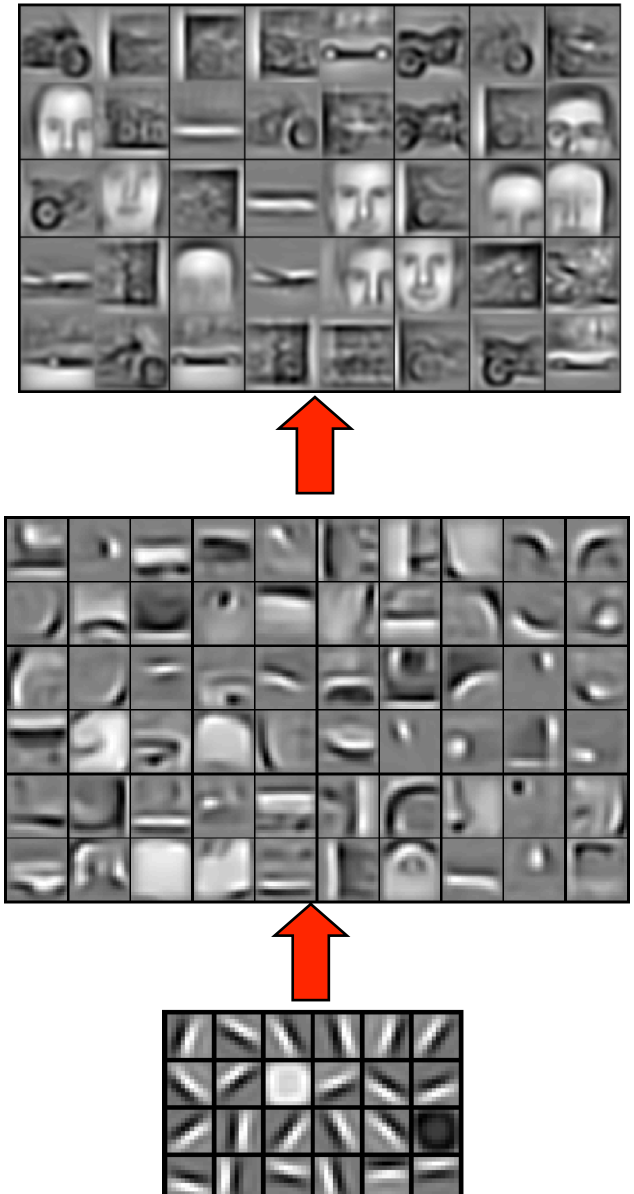
Based on materials
by Andrew Ng

Learning of Object Parts



Slide credit: Andrew Ng

Training on Multiple Objects



Trained on 4 classes (cars, faces, motorbikes, airplanes).

Second layer: Shared-features and object-specific features.

Third layer: More specific features.

Slide credit: Andrew Ng

Scene Labeling via Deep Learning

[Farabet et al. ICML 2012, PAMI 2013]

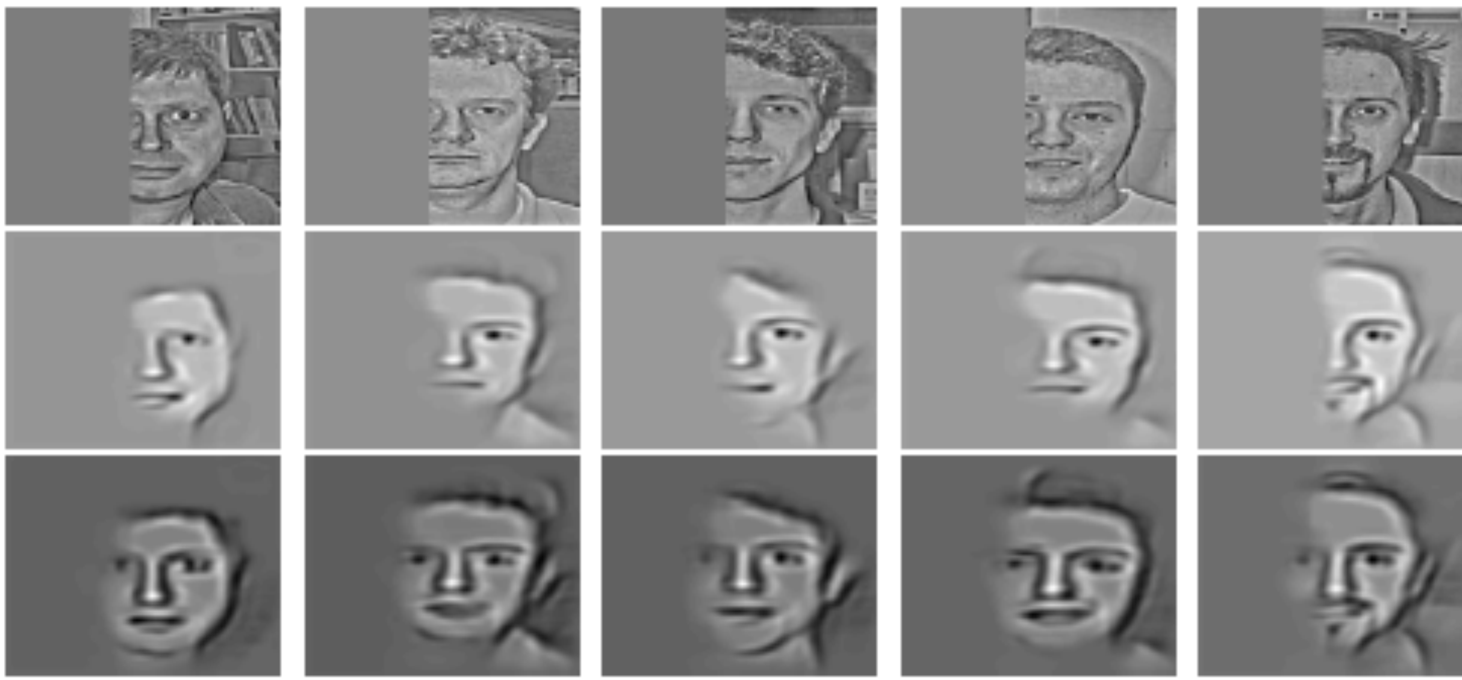
Inference from Deep Learned Models

Generating posterior samples from faces by "filling in" experiments
(cf. Lee and Mumford, 2003). Combine bottom-up and top-down inference.

Input images

Samples from feedforward Inference (control)

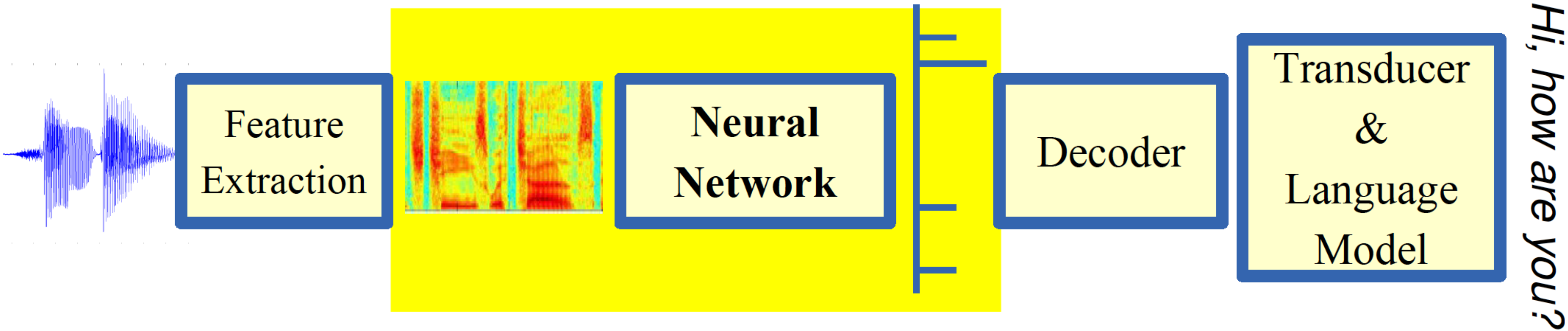
Samples from Full posterior inference



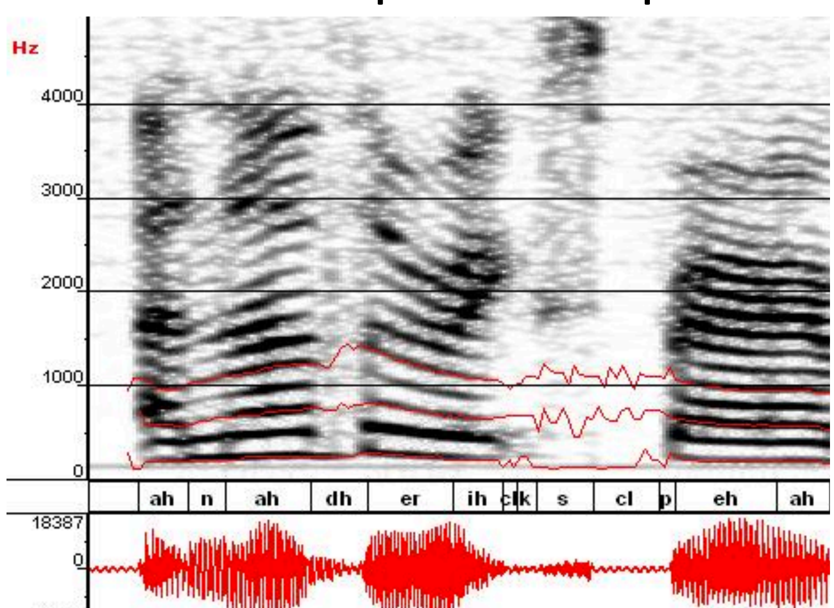
Slide credit: Andrew Ng

Machine Learning in Automatic Speech Recognition

A Typical Speech Recognition System



ML used to predict of phone states from the sound spectrogram



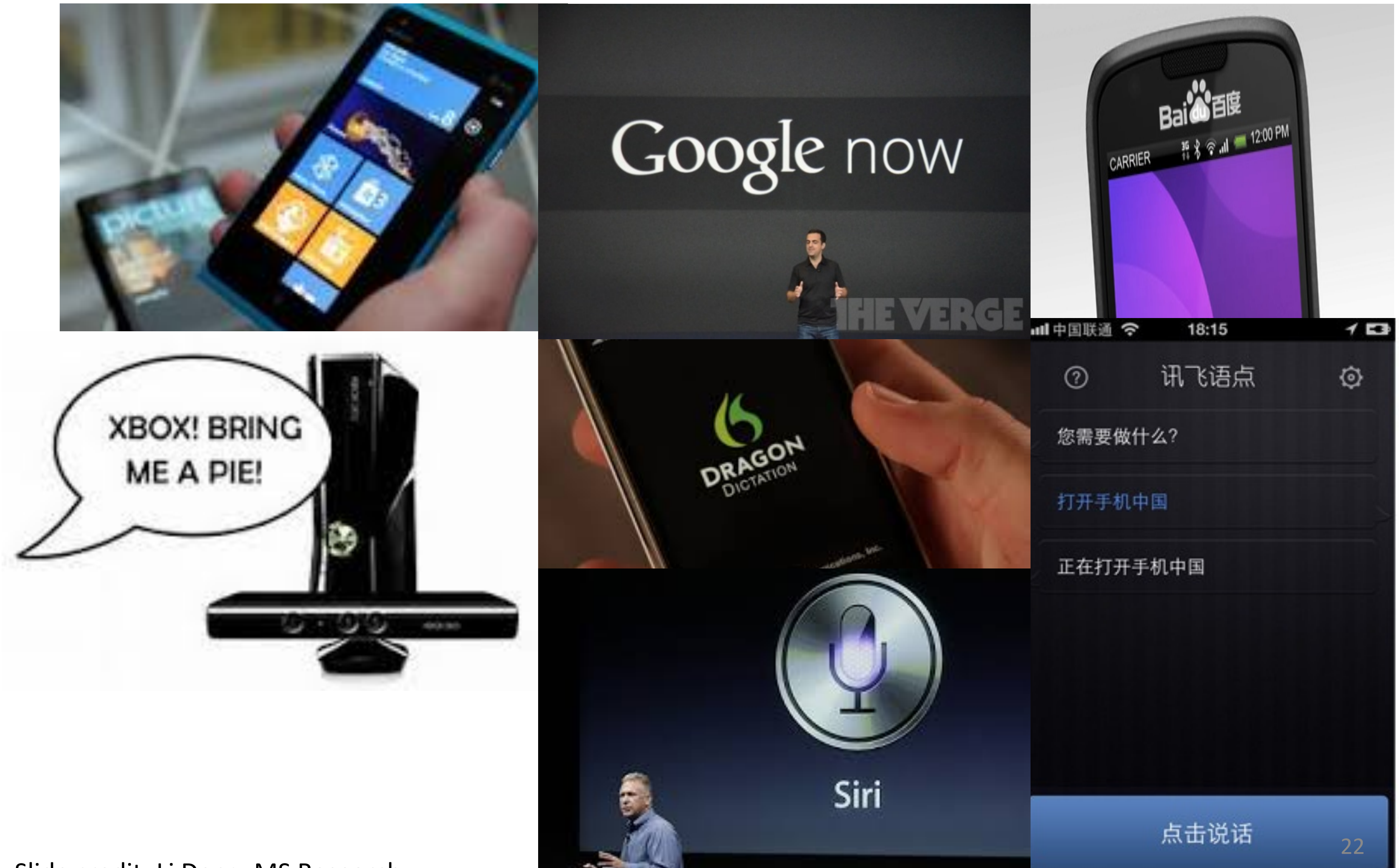
Deep learning has state-of-the-art results

# Hidden Layers	1	2	4	8	10	12
Word Error Rate %	16.0	12.8	11.4	10.9	11.0	11.1

Baseline GMM performance = 15.4%

[Zeiler et al. “On rectified linear units for speech recognition” ICASSP 2013]

Impact of Deep Learning in Speech Technology



Slide credit: Li Deng, MS Research

Types of Learning

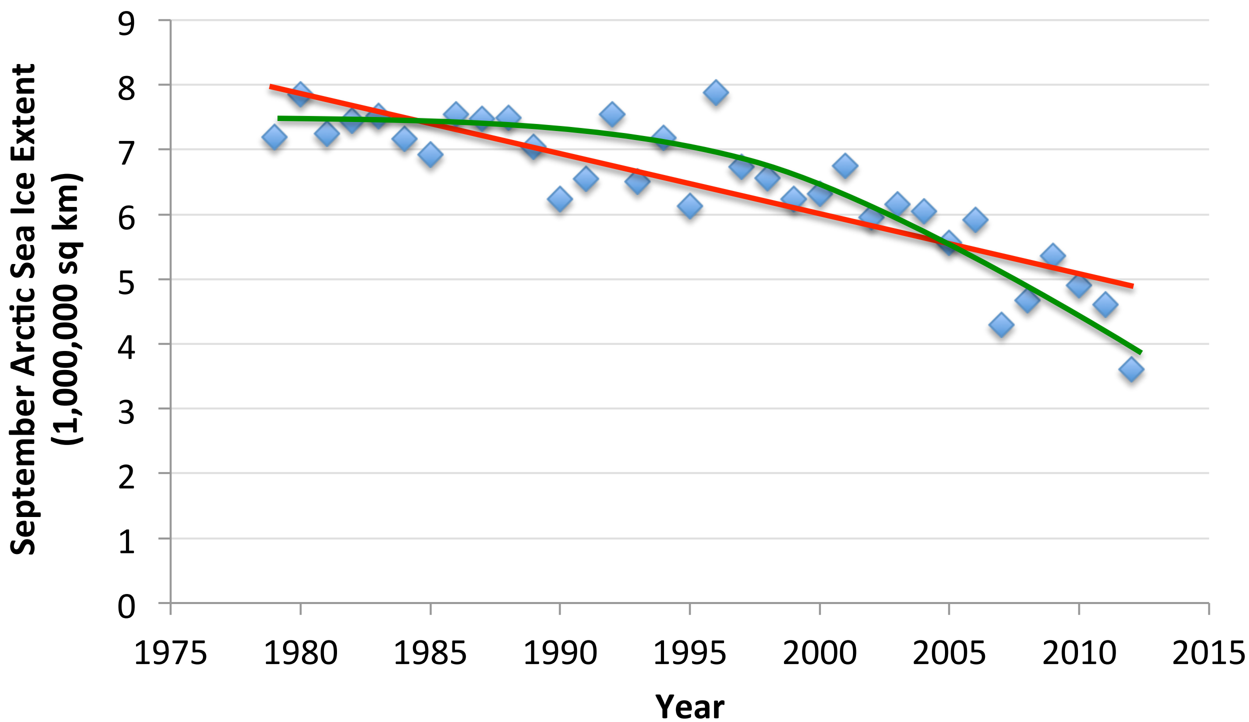
Types of Learning

- **Supervised (inductive) learning**
  – Given: training data + desired outputs (labels)
- **Unsupervised learning**
  – Given: training data (without desired outputs)
- **Semi-supervised learning**
  – Given: training data + a few desired outputs
- **Reinforcement learning**
  – Rewards from sequence of actions

Based on slide by Pedro Domingos

Supervised Learning: Regression

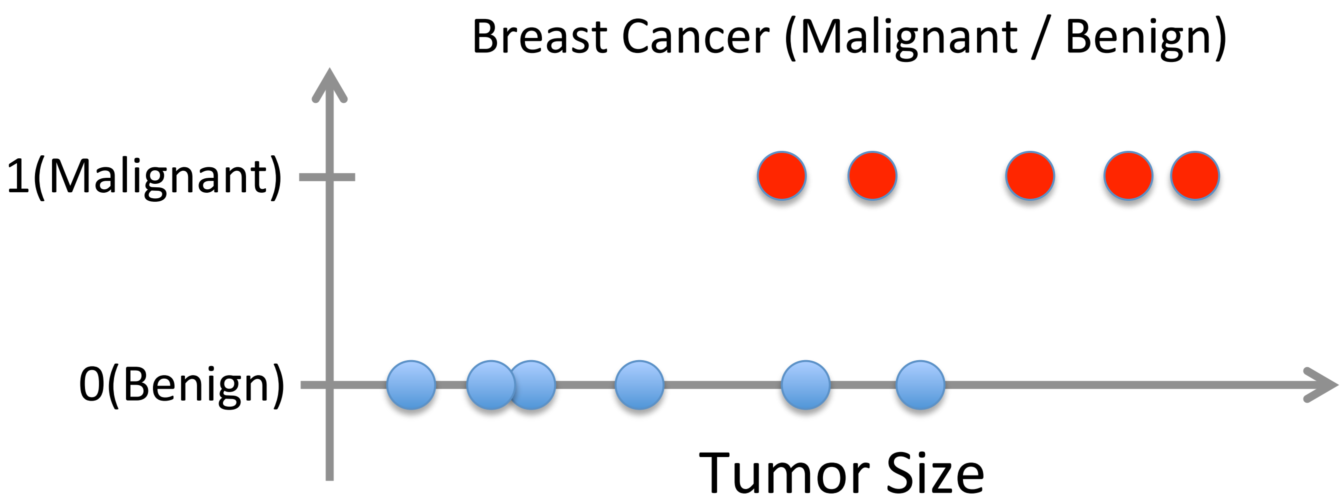
- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is real-valued == regression



Data from G. Witt. Journal of Statistics Education, Volume 21, Number 1 (2013)

Supervised Learning: Classification

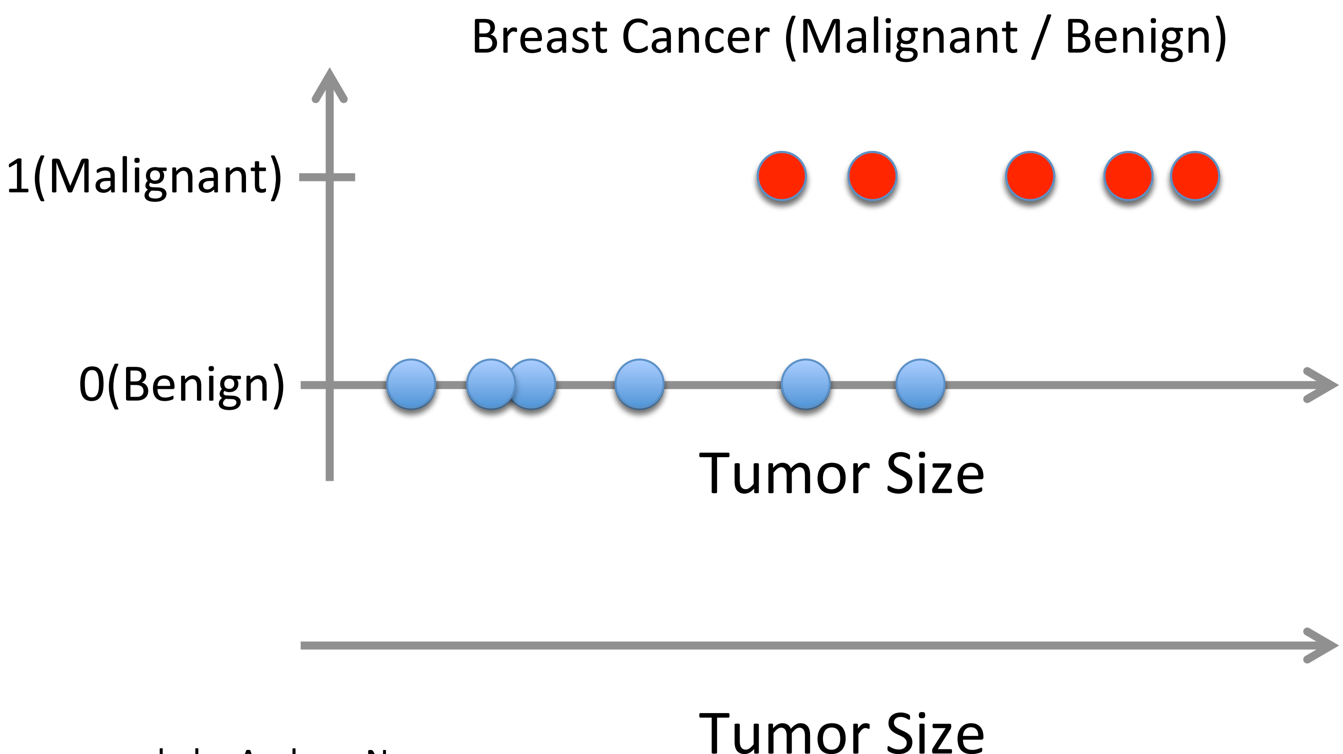
- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



Based on example by Andrew Ng

Supervised Learning: Classification

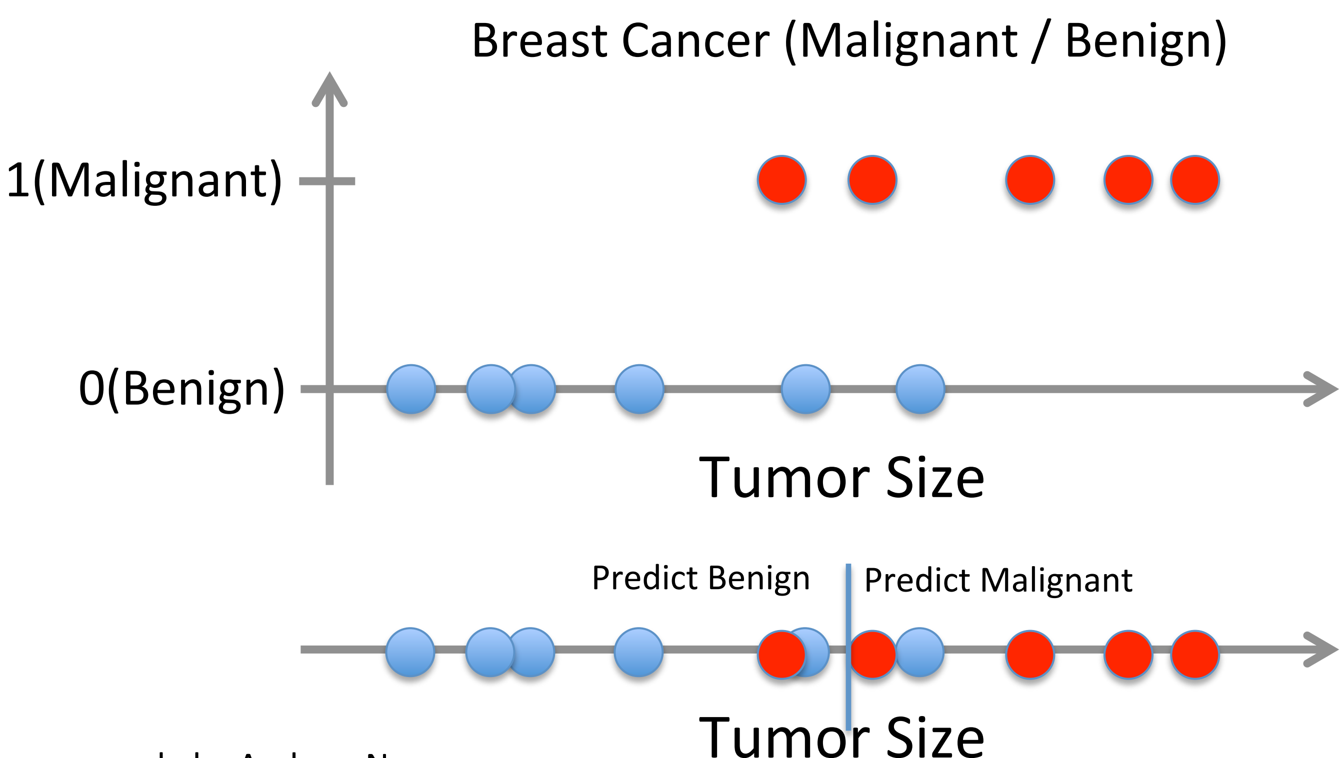
- Given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



Based on example by Andrew Ng

Supervised Learning: Classification

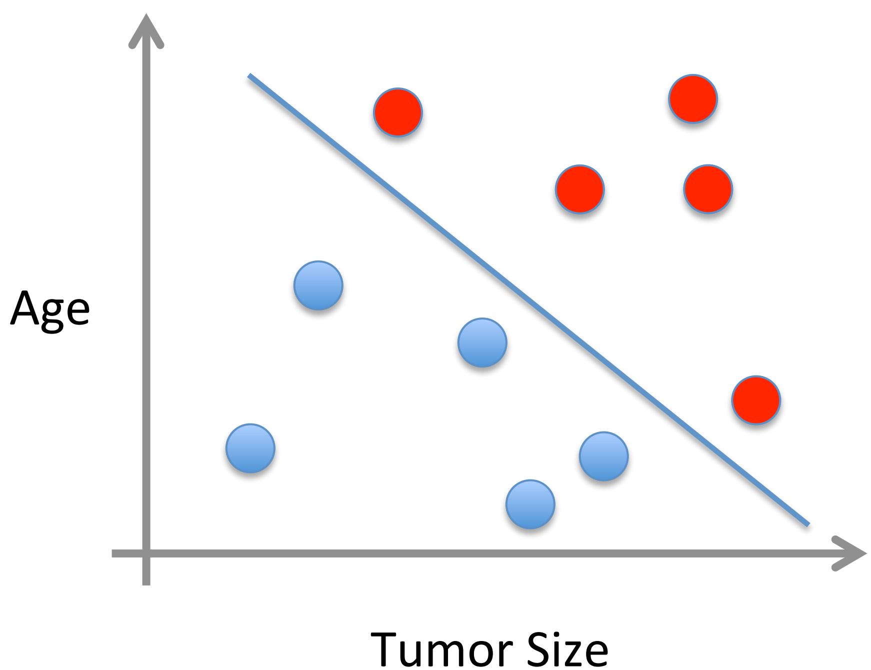
- Given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is categorical == classification



Based on example by Andrew Ng

Supervised Learning

- x can be multi-dimensional
  - Each dimension corresponds to an attribute



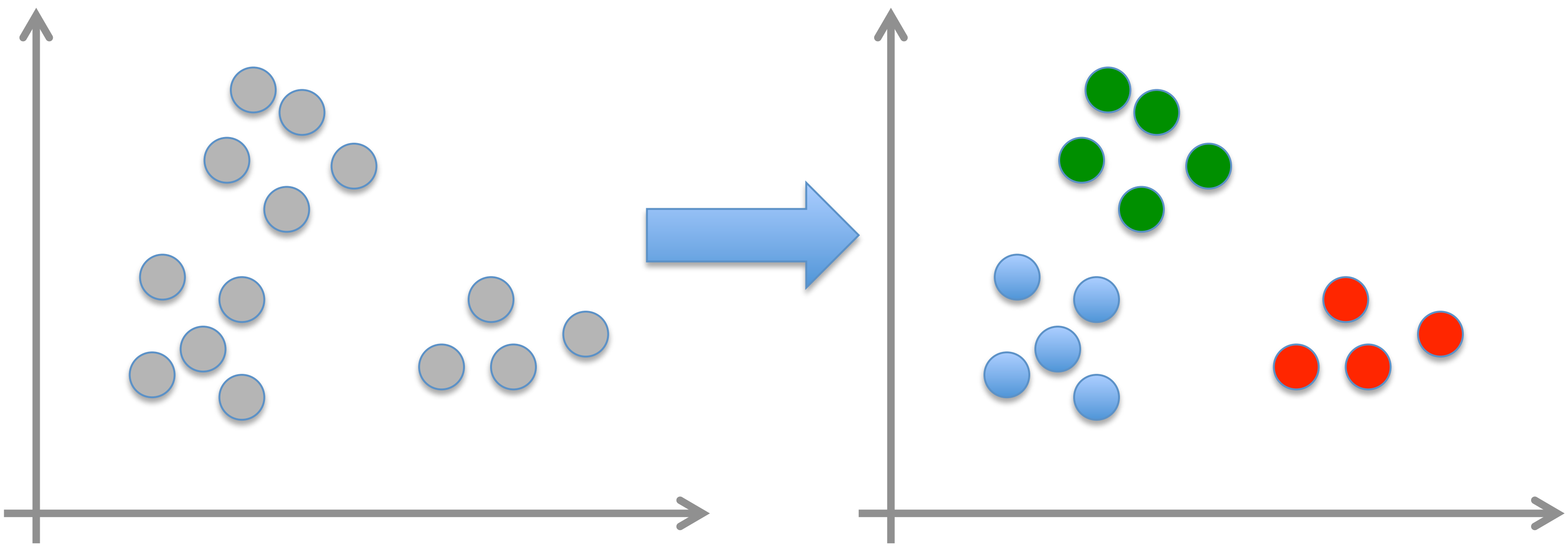
- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

…

Based on example by Andrew Ng

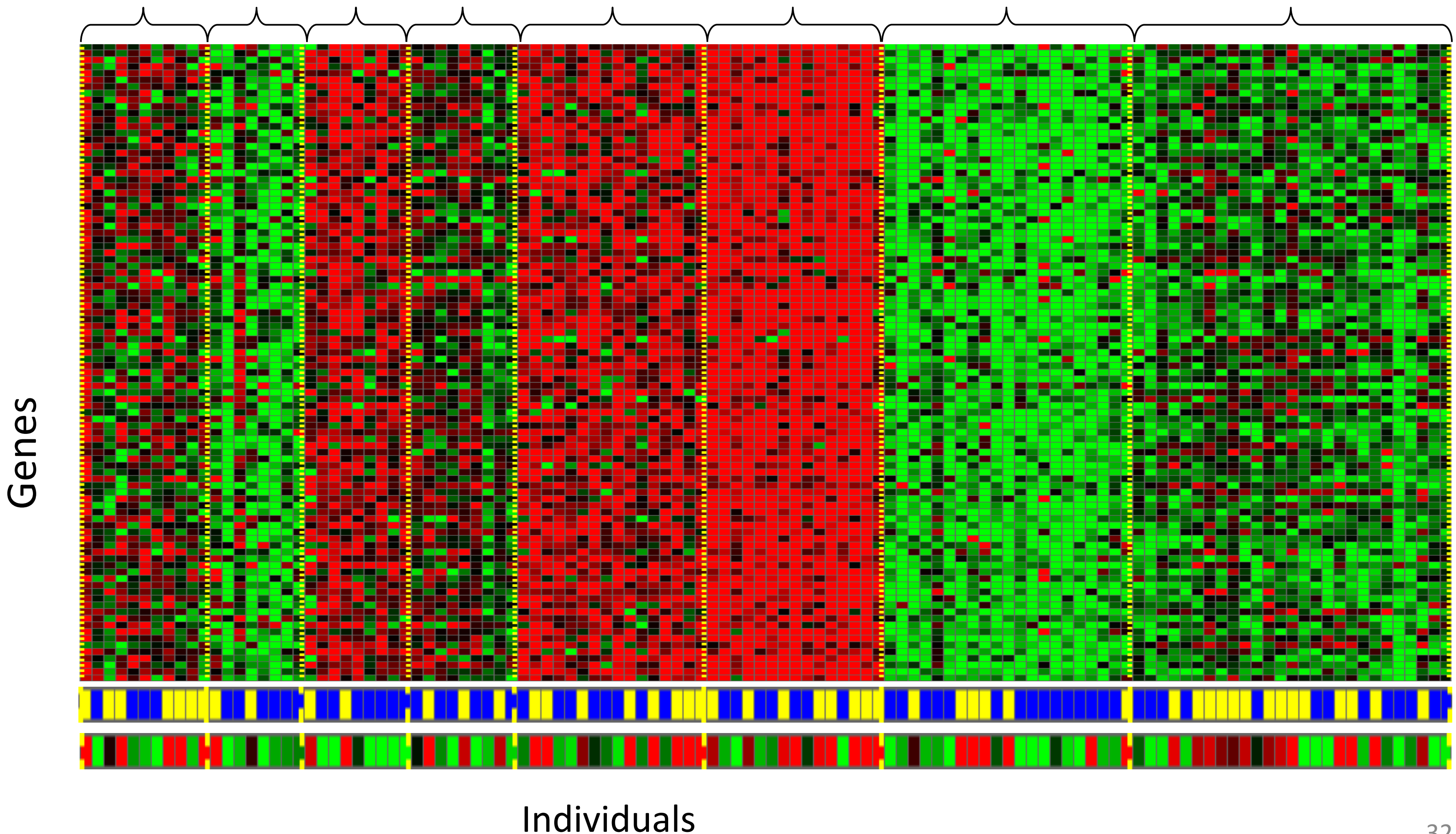
Unsupervised Learning

- Given x_1, x_2, ..., x_n (without labels)
- Output hidden structure behind the x's
  - E.g., clustering



Unsupervised Learning

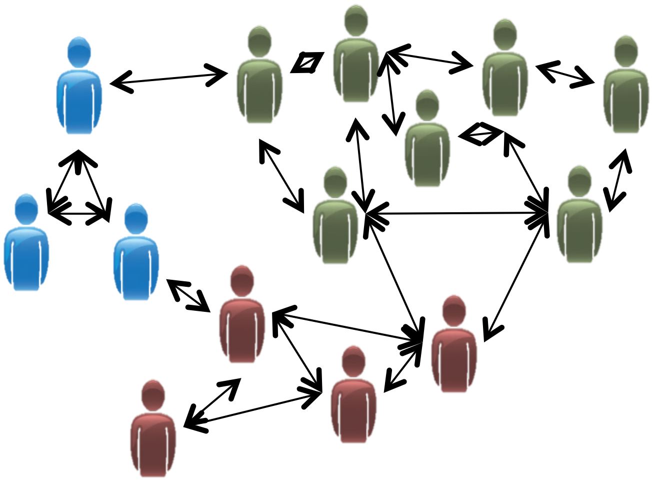
Genomics application: group individuals by genetic similarity



[Source: Daphne Koller]

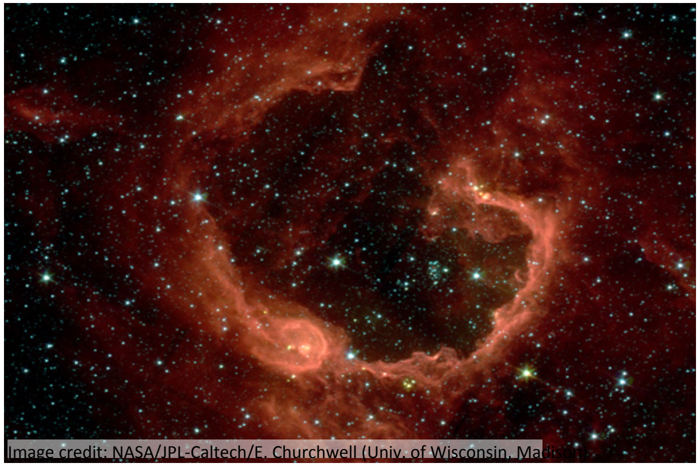
Unsupervised Learning

Organize computing clusters



Social network analysis

Market segmentation



Astronomical data analysis

Slide credit: Andrew Ng

Unsupervised Learning

- Independent component analysis – separate a combined signal into its original sources

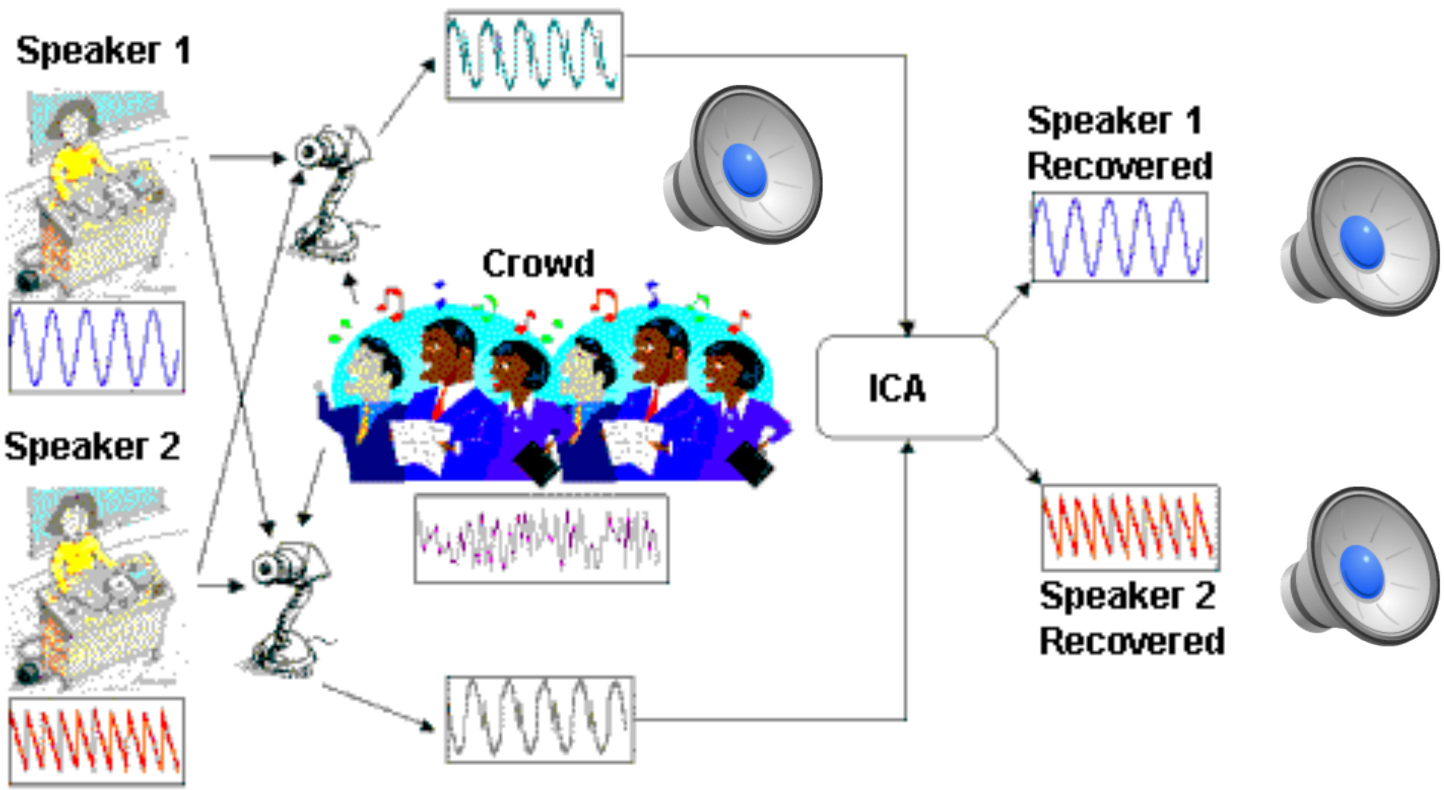


Image credit: statsoft.com Audio from http://www.ism.ac.jp/~shiro/research/blindsep.html

Unsupervised Learning

- Independent component analysis – separate a combined signal into its original sources

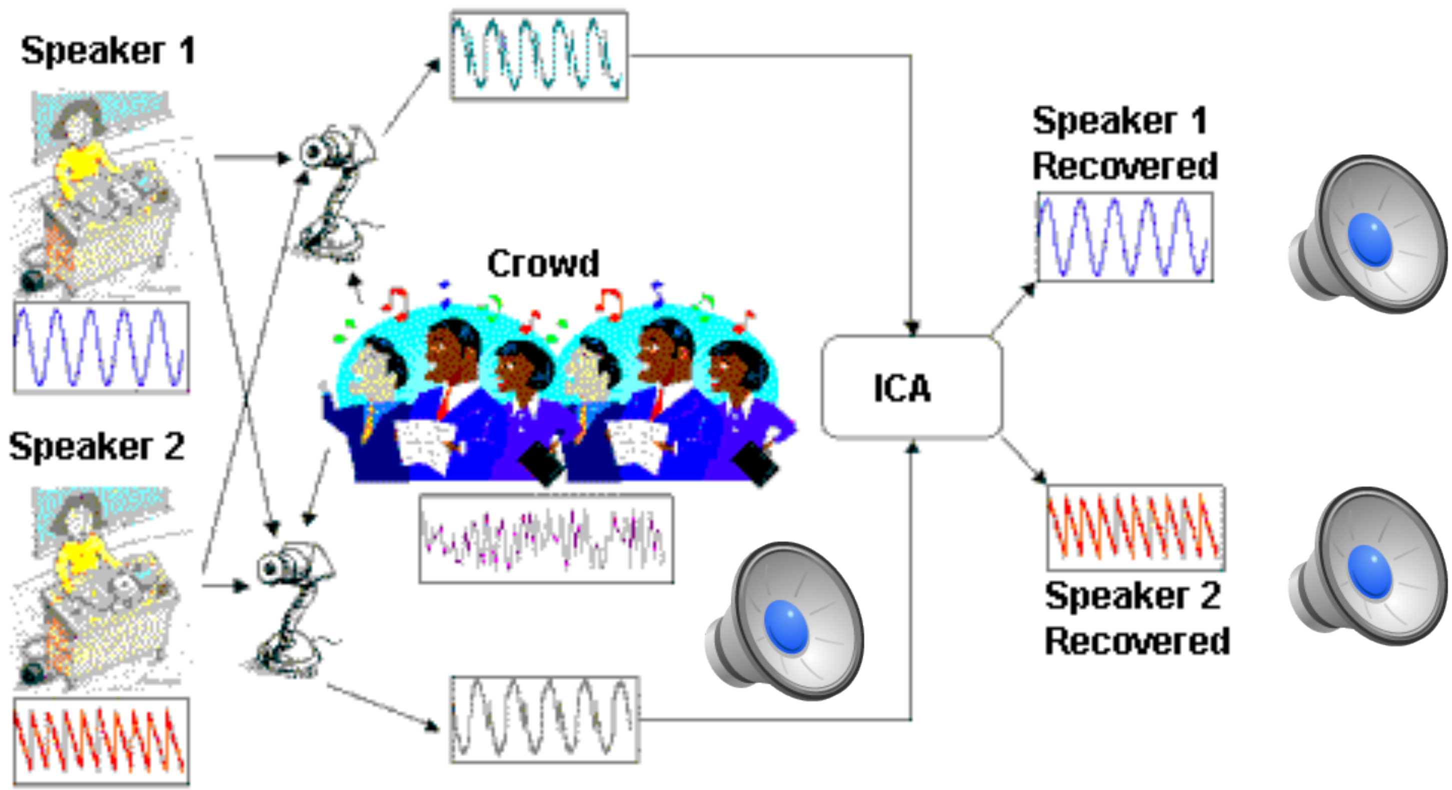
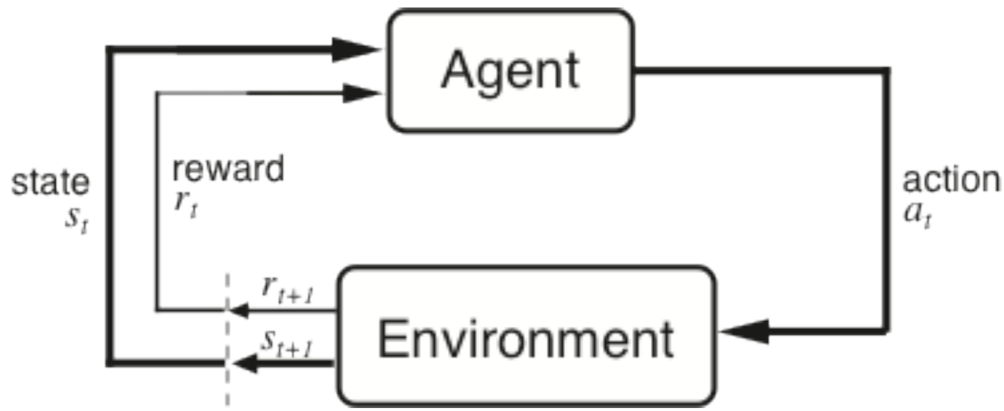


Image credit: statsoft.com Audio from http://www.ism.ac.jp/~shiro/research/blindsep.html

Reinforcement Learning

- Given a sequence of states and actions with (delayed) rewards, output a policy
  - Policy is a mapping from states → actions that tells you what to do in a given state
- Examples:
  - Credit assignment problem
  - Game playing
  - Robot in a maze
  - Balance a pole on your hand

The Agent-Environment Interface



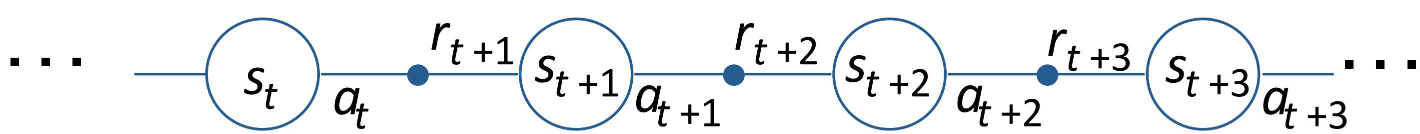
Agent and environment interact at discrete time steps : $t = 0, 1, 2, K$

Agent observes state at step t: $s_t \in S$

produces action at step t: $a_t \in A(s_t)$

gets resulting reward : $r_{t+1} \in \Re$

and resulting next state : s_{t+1}



Slide credit: Sutton & Barto

Reinforcement Learning

https://www.youtube.com/watch?v=4cgWya-wjgY

Inverse Reinforcement Learning

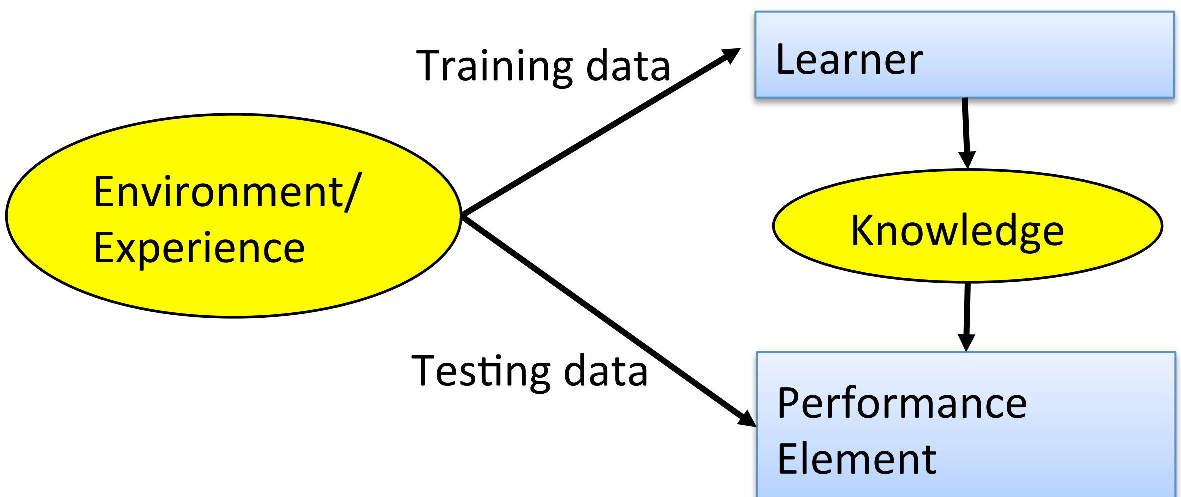
- Learn policy from user demonstrations

Stanford Autonomous Helicopter
http://heli.stanford.edu/
https://www.youtube.com/watch?v=VCdxqn0fcnE

Framing a Learning Problem

Designing a Learning System

- Choose the training experience
- Choose exactly what is to be learned
  - i.e. the ***target function***
- Choose how to represent the target function
- Choose a learning algorithm to infer the target function from the experience



Based on slide by Ray Mooney

Training vs. Test Distribution

- We generally assume that the training and test examples are independently drawn from the same overall distribution of data
  - We call this "i.i.d" which stands for "independent and identically distributed"

- If examples are not independent, requires ***collective classification***

- If test distribution is different, requires ***transfer learning***

Slide credit: Ray Mooney

ML in a Nutshell

- Tens of thousands of machine learning algorithms
  - Hundreds new every year

- Every ML algorithm has three components:
  - **Representation**
  - **Optimization**
  - **Evaluation**

Slide credit: Pedro Domingos

Various Function Representations

- Numerical functions
  - Linear regression
  - Neural networks
  - Support vector machines
- Symbolic functions
  - Decision trees
  - Rules in propositional logic
  - Rules in first-order predicate logic
- Instance-based functions
  - Nearest-neighbor
  - Case-based
- Probabilistic Graphical Models
  - Naïve Bayes
  - Bayesian networks
  - Hidden-Markov Models (HMMs)
  - Probabilistic Context Free Grammars (PCFGs)
  - Markov networks

Slide credit: Ray Mooney

Various Search/Optimization Algorithms

- Gradient descent
  - Perceptron
  - Backpropagation
- Dynamic Programming
  - HMM Learning
  - PCFG Learning
- Divide and Conquer
  - Decision tree induction
  - Rule learning
- Evolutionary Computation
  - Genetic Algorithms (GAs)
  - Genetic Programming (GP)
  - Neuro-evolution

Slide credit: Ray Mooney

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- etc.

Slide credit: Pedro Domingos

ML in Practice

Loop

- Understand domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learn models
- Interpret results
- Consolidate and deploy discovered knowledge

Based on a slide by Pedro Domingos

Lessons Learned about Learning

- Learning can be viewed as using direct or indirect experience to approximate a chosen target function.

- Function approximation can be viewed as a search through a space of hypotheses (representations of functions) for one that best fits a set of training data.

- Different learning methods assume different hypothesis spaces (representation languages) and/or employ different search techniques.

Slide credit: Ray Mooney

A Brief History of Machine Learning

History of Machine Learning

- 1950s
  - Samuel's checker player
  - Selfridge's Pandemonium
- 1960s:
  - Neural networks: Perceptron
  - Pattern recognition
  - Learning in the limit theory
  - Minsky and Papert prove limitations of Perceptron
- 1970s:
  - Symbolic concept induction
  - Winston's arch learner
  - Expert systems and the knowledge acquisition bottleneck
  - Quinlan's ID3
  - Michalski's AQ and soybean diagnosis
  - Scientific discovery with BACON
  - Mathematical discovery with AM

Slide credit: Ray Mooney

History of Machine Learning (cont.)

- 1980s:
  - Advanced decision tree and rule learning
  - Explanation-based Learning (EBL)
  - Learning and planning and problem solving
  - Utility problem
  - Analogy
  - Cognitive architectures
  - Resurgence of neural networks (connectionism, backpropagation)
  - Valiant's PAC Learning Theory
  - Focus on experimental methodology
- 1990s
  - Data mining
  - Adaptive software agents and web applications
  - Text learning
  - Reinforcement learning (RL)
  - Inductive Logic Programming (ILP)
  - Ensembles: Bagging, Boosting, and Stacking
  - Bayes Net learning

Slide credit: Ray Mooney

History of Machine Learning (cont.)

- 2000s
  - Support vector machines & kernel methods
  - Graphical models
  - Statistical relational learning
  - Transfer learning
  - Sequence labeling
  - Collective classification and structured outputs
  - Computer Systems Applications (Compilers, Debugging, Graphics, Security)
  - E-mail management
  - Personalized assistants that learn
  - Learning in robotics and vision
- 2010s
  - Deep learning systems
  - Learning for big data
  - Bayesian methods
  - Multi-task & lifelong learning
  - Applications to vision, speech, social networks, learning to read, etc.
  - ???

Based on slide by Ray Mooney

What We'll Cover in this Course

- **Supervised learning**
  - Decision tree induction
  - Linear regression
  - Logistic regression
  - Support vector machines & kernel methods
  - Model ensembles
  - Bayesian learning
  - Neural networks & deep learning
  - Learning theory
- **Unsupervised learning**
  - Clustering
  - Dimensionality reduction
- **Reinforcement learning**
  - Temporal difference learning
  - Q learning
- **Evaluation**
- **Applications**

Our focus will be on applying machine learning to real applications

Extra Material

Sample Learning Problem

Learn to play checkers from self-play

Learn to play checkers from self-play

Our Goal: Develop an approach analogous to that used in the first machine learning system developed by Arthur Samuels at IBM in 1959.

Based on example by Ray Mooney

What **Training Experience** will we have?

- Direct experience: Given sample input and output pairs for a useful target function.
  - Checker boards labeled with the correct move, e.g. extracted from record of expert play
- Indirect experience: Given feedback which is ***not*** direct I/O pairs for a useful target function.
  - Potentially arbitrary sequences of game moves and their final game results.
- Credit/Blame Assignment Problem: How do we assign credit/blame to individual moves given only indirect feedback?

Based on example by Ray Mooney

Potential Sources of Training Data

- Provided random examples outside of the learner's control
  - Are negative examples available or only positive?
- Good training examples selected by a "benevolent teacher"
  - e.g., "Near miss" examples
- We could make the learner "active":
  - Learner can query an oracle about label of an unlabeled example in the environment
  - Learner can construct an arbitrary example and query an oracle for its label
  - Learner can design and run experiments directly in the environment without any human guidance

Slide credit: Ray Mooney

Choosing a Target Function

- What function is to be learned and how will it be used by the performance system?
- For checkers, assume we are given a function for generating the legal moves for a given board position
- We then want to decide the best move:
  - Could learn a function:

    ChooseMove(board, legal-moves) → best-move
  - Or could learn an ***evaluation function*** *V*(board) → $\mathbb{R}$ that gives each board position a score for how favorable it is
    - *V* can be used to pick a move by applying each legal move, scoring the resulting board position, and choosing the move that results in the highest scoring board position.

Slide credit: Ray Mooney

Ideal Definition of *V*(*b*)

- If *b* is a final winning board, then *V*(*b*) = 100
- If *b* is a final losing board, then *V*(*b*) = –100
- If *b* is a final draw board, then *V*(*b*) = 0
- Otherwise, *V*(*b*) = *V*(*b*´), where *b*´ is the highest scoring final board position that is achieved starting from *b* and playing optimally until the end of the game (assuming the opponent plays optimally as well)
  - Can be computed using complete mini-max search of the finite game tree.

Slide credit: Ray Mooney

Approximating $V(b)$

- Problem: Computing $V(b)$ is intractable since it involves searching the complete exponential game tree.

- Solution: We need to use an *approximation* to the ideal evaluation function that can be computed in reasonable (polynomial) time

Based on slide by Ray Mooney

Representing the Target Function

- Target function can be represented in many ways:
  - lookup table, symbolic rules, numerical function, neural network, etc.

- There is a trade-off between the expressiveness of a representation and the ease of learning
  - The more expressive a representation, the better it will be at approximating an arbitrary function
  - ...but, the more examples will be needed to learn an accurate function

Slide credit: Ray Mooney

Linear Function for Representing *V*(*b*)

- In checkers, use a linear approximation of the evaluation function.

$$\widehat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

  - *bp*(*b*): # black pieces on board b
  - *rp*(*b*): # red pieces on board b
  - *bk*(*b*): #black kings on board b
  - *rk*(*b*): # red kings on board b
  - *bt*(*b*): # black pieces threatened

    (i.e. which can be immediately taken by red on its next turn)
  - *rt*(*b*): # red pieces threatened

Slide credit: Ray Mooney

Obtaining Training Values

- Direct supervision may be available for the target function.

  E.g., a labeled instance such as:

  $$\Big(\underbrace{\langle bp = 3, rp = 0, bk = 1, rk = 0, bt = 0, rt = 0\rangle}_{\boldsymbol{x}},\ \underbrace{100}_{y}\Big)$$ (black wins)

- With indirect feedback, training values can be estimated using ***temporal difference learning*** (used in ***reinforcement learning*** where supervision is ***delayed reward***)

Slide credit: Ray Mooney

Learning Algorithm

- Uses training values for the target function to:
  - induce a hypothesized definition that fits these examples
  - ...and (hopefully) generalizes to unseen examples.

- Attempts to minimize some measure of error (***loss function***) such as ***mean squared error***:

$$E = \frac{1}{|B|} \sum_{b \in B} \Big(\underbrace{V_{train}(b)}_{\text{Given label}} - \underbrace{\hat{V}(b)}_{\text{Predicted label}} \Big)^2$$

Based on slide by Ray Mooney

Least Mean Squares (LMS) Algorithm

- A gradient descent algorithm that incrementally updates the weights of a linear function in an attempt to minimize the mean squared error

initialize weights randomly

loop until weights converge :

for each training example *b* do :

1) Compute the absolute error :

$$error(b) = V_{train}(b) - \widehat{V}(b)$$

2) For each board feature f_i update its weight w_i :

$$w_i = w_i + c \cdot f_i \cdot error(b)$$

for some small constant (learning rate) c

Slide credit: Ray Mooney

LMS Discussion

- Intuitively, LMS executes the following rules:
  - If the output for an example is correct, make no change.
  - If the output is too high, lower the weights proportional to the values of their corresponding features, so the overall output decreases.
  - If the output is too low, increase the weights proportional to the values of their corresponding features, so the overall output increases.
- Under the proper weak assumptions, LMS can be proven to eventually converge to a set of weights that minimizes the mean squared error.

Slide credit: Ray Mooney