Lecture #3: Machine Learning

Janardhan Rao (Jana) Doppa

School of EECS, Washington State University

Machine Learning is Everywhere

- "If you invent a breakthrough in artificial intelligence, so machines can learn," Mr. Gates responded, "that is worth 10 Microsofts."

(Quoted in NY Times, Monday March 3, 2004)

Machine Learning is Everywhere

- Spam filtering

googleteam GOOGLE LOTTERY WINNER! CONTACT

From: googleteam **To:**
Subject: GOOGLE LOTTERY WINNER! CONTACT YOUR AGENT TO CLAIM YOUR PRIZE.

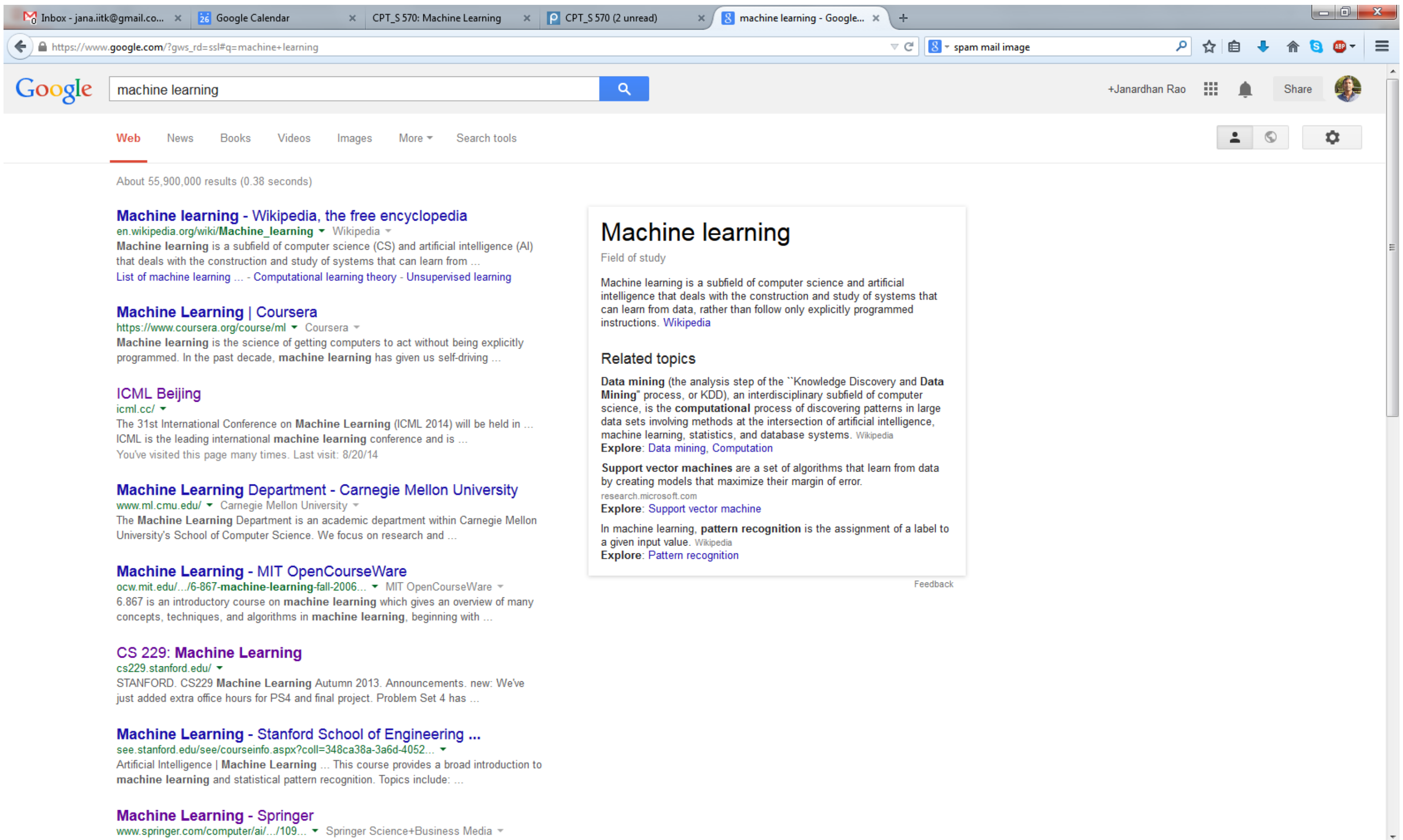
GOOGLE LOTTERY INTERNATIONAL
INTERNATIONAL PROMOTION / PRIZE AWARD .
(WE ENCOURAGE GLOBALIZATION)
FROM: THE LOTTERY COORDINATOR,
GOOGLE B.V. 44 9459 PE.
RESULTS FOR CATEGORY "A" DRAWS
Congratulations to you as we bring to your notice, the results of the First Ca
inform you that your email address have emerged a winner of One Million (1,
money of Two Million (2,000,000.00) Euro shared among the 2 winners in thi
email addresses of individuals and companies from Africa, America, Asia, A
CONGRATULATIONS!
Your fund is now deposited with the paying Bank. In your best interest to avo
award strictly from public notice until the process of transferring your claims
NOTE: to file for your claim, please contact the claim department below on e
****

Machine Learning is Everywhere

- Optical Character Recognition (OCR)

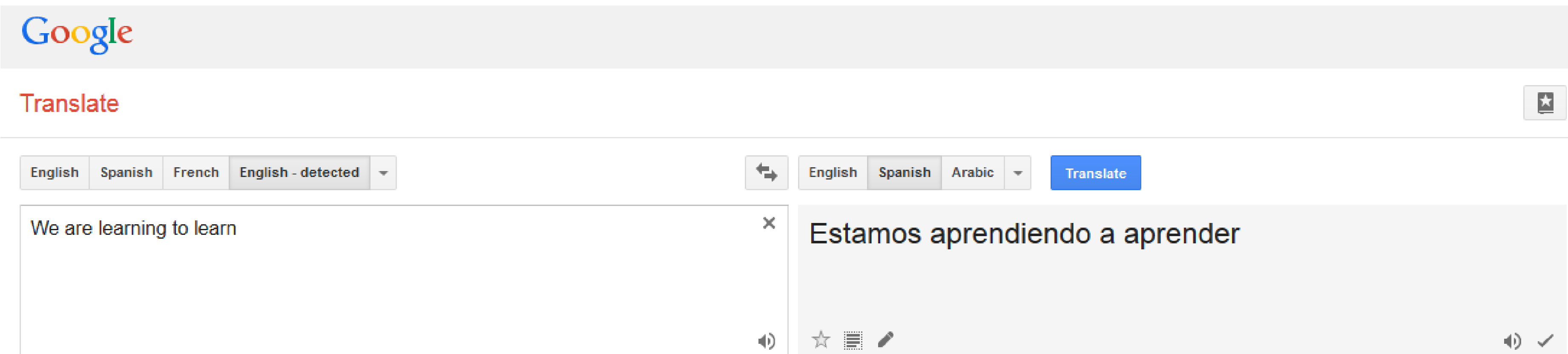
Machine Learning is Everywhere

- Search engines



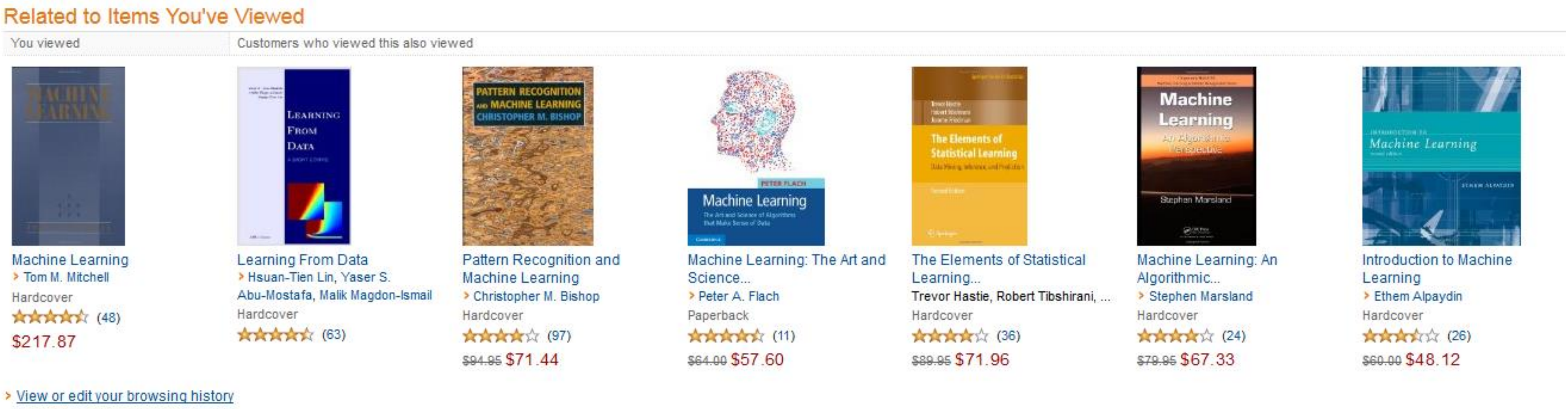
Machine Learning is Everywhere

- Automatic Translation



Machine Learning is Everywhere

- Recommendation Engines



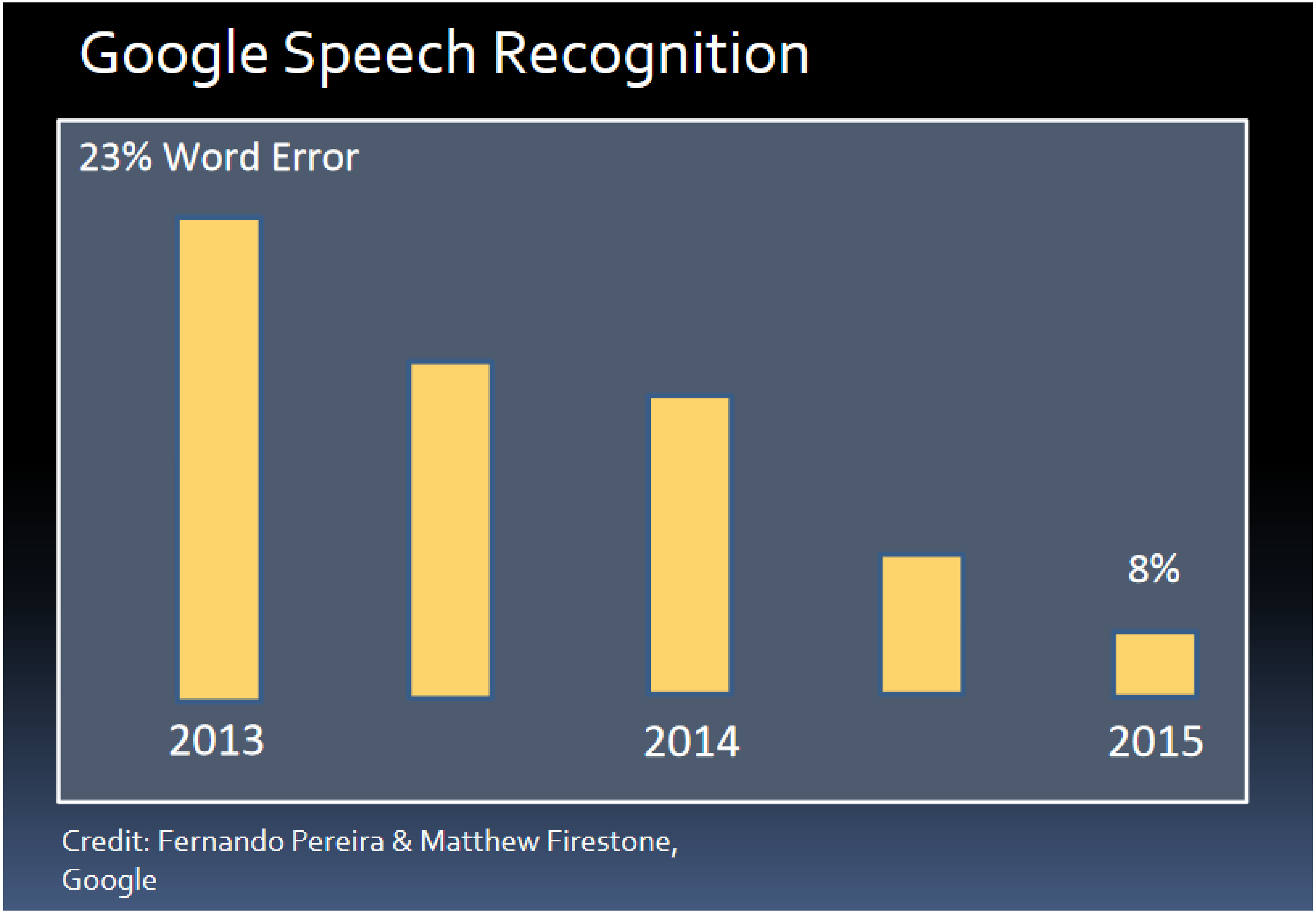
Machine Learning is Everywhere

- Self-driving cars

Google's Self Driving Car for Blind People

by EDITORS on Apr 6, 2012 · 4:07 pm

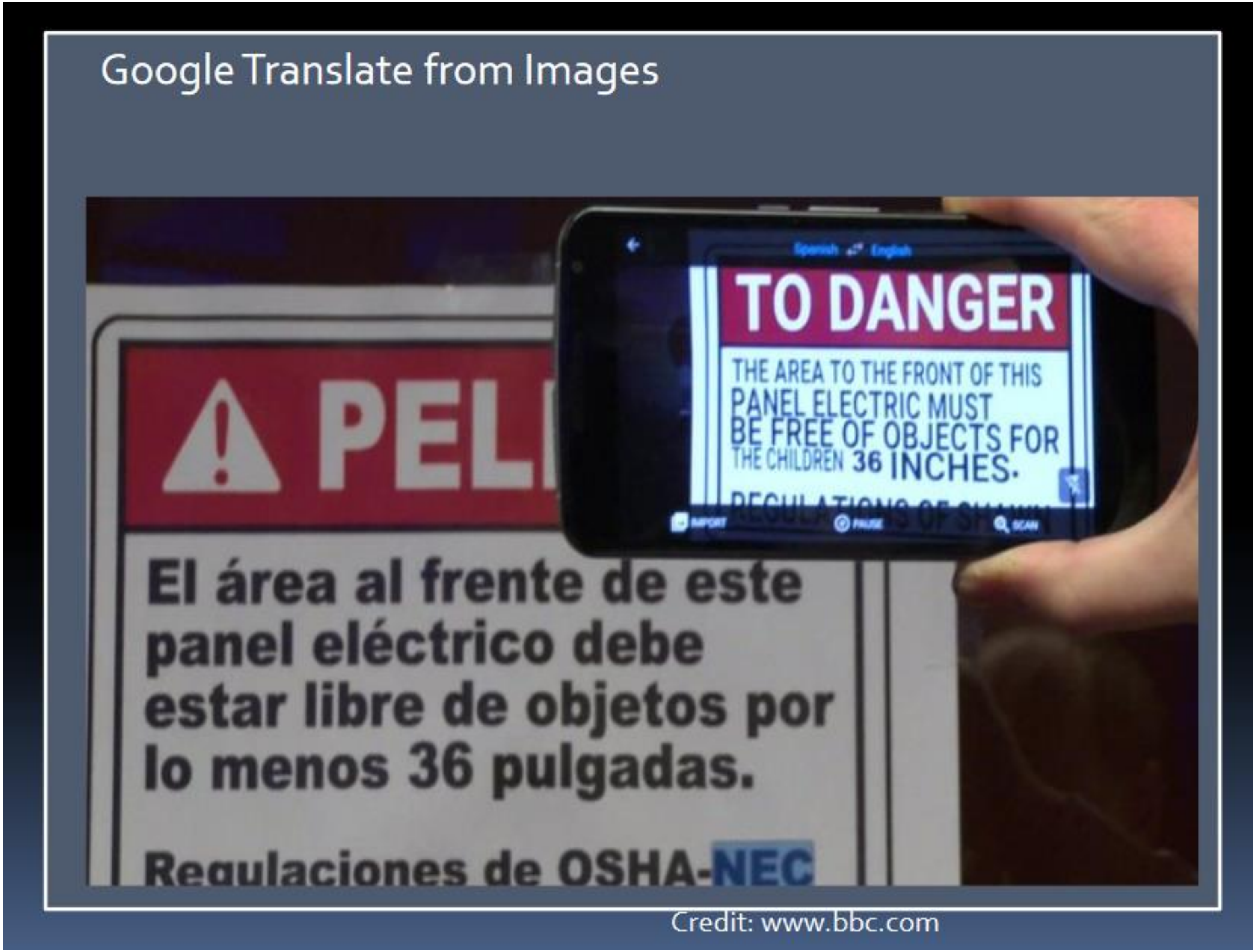
ML Successes: Perception



Credit: Tom Dietterich

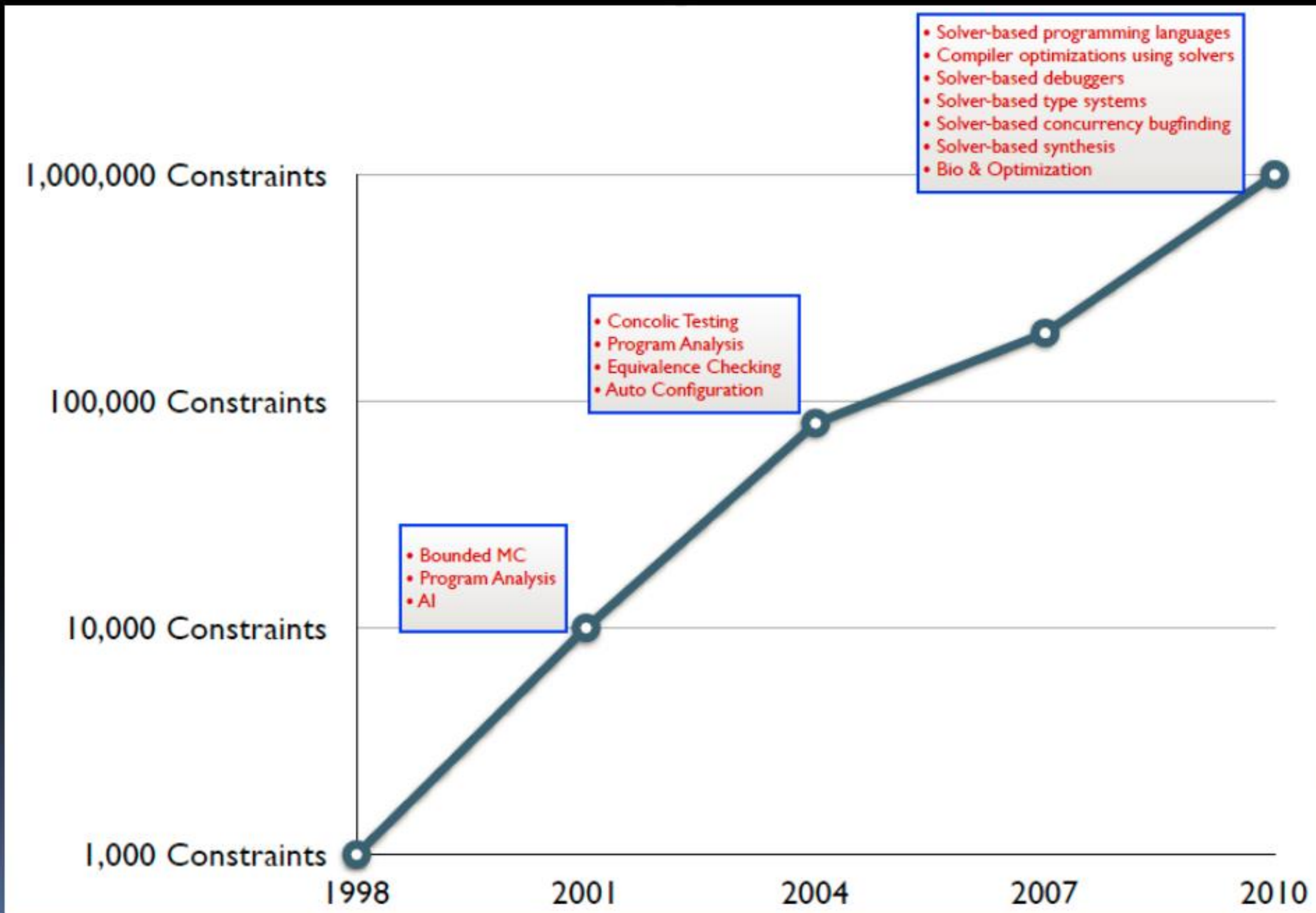
ML Successes: Image Captioning

ML Successes: Perception + Translation



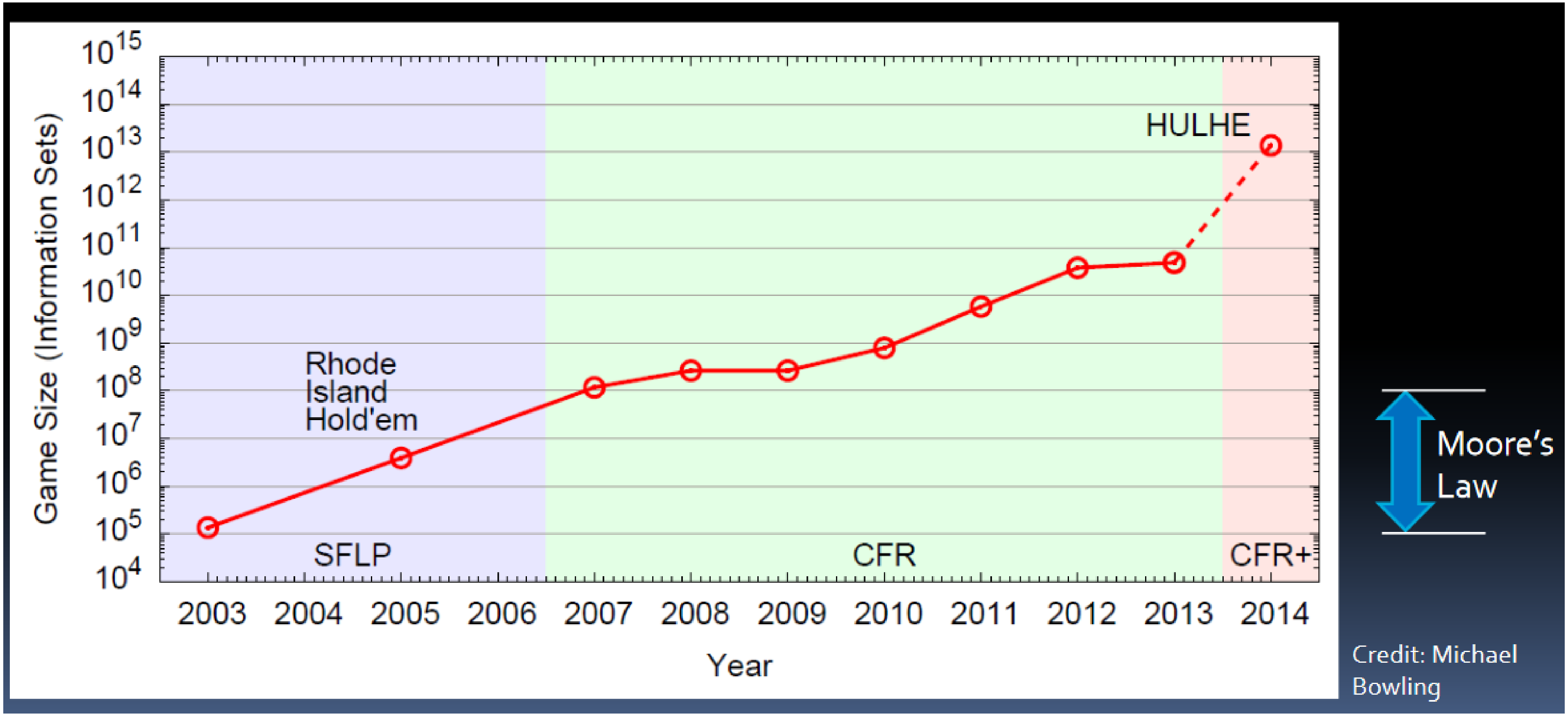
ML Successes: Skype Translator

ML Successes: Reasoning (SAT)

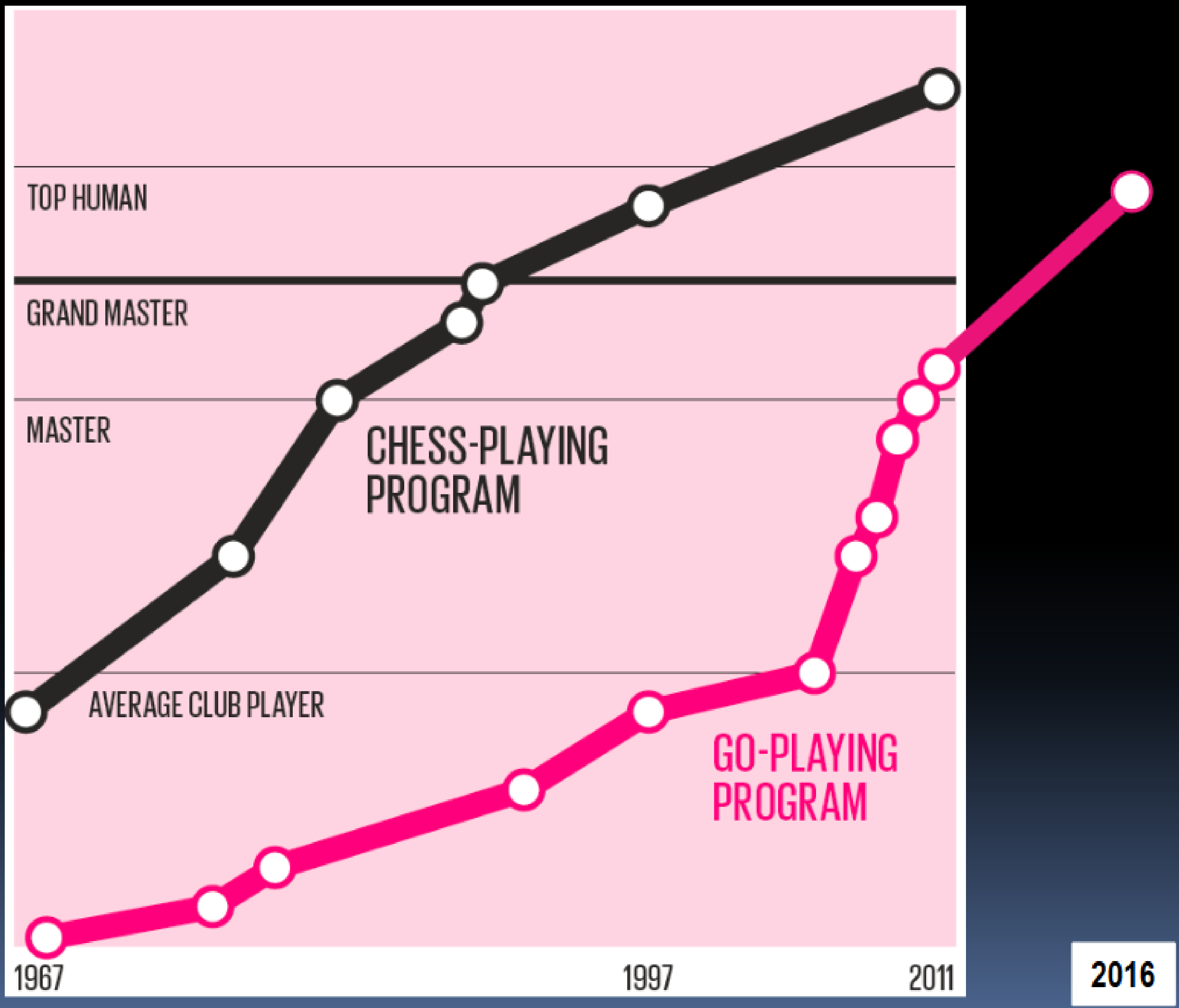


Credit: Vijay Ganesh

ML Successes: Poker



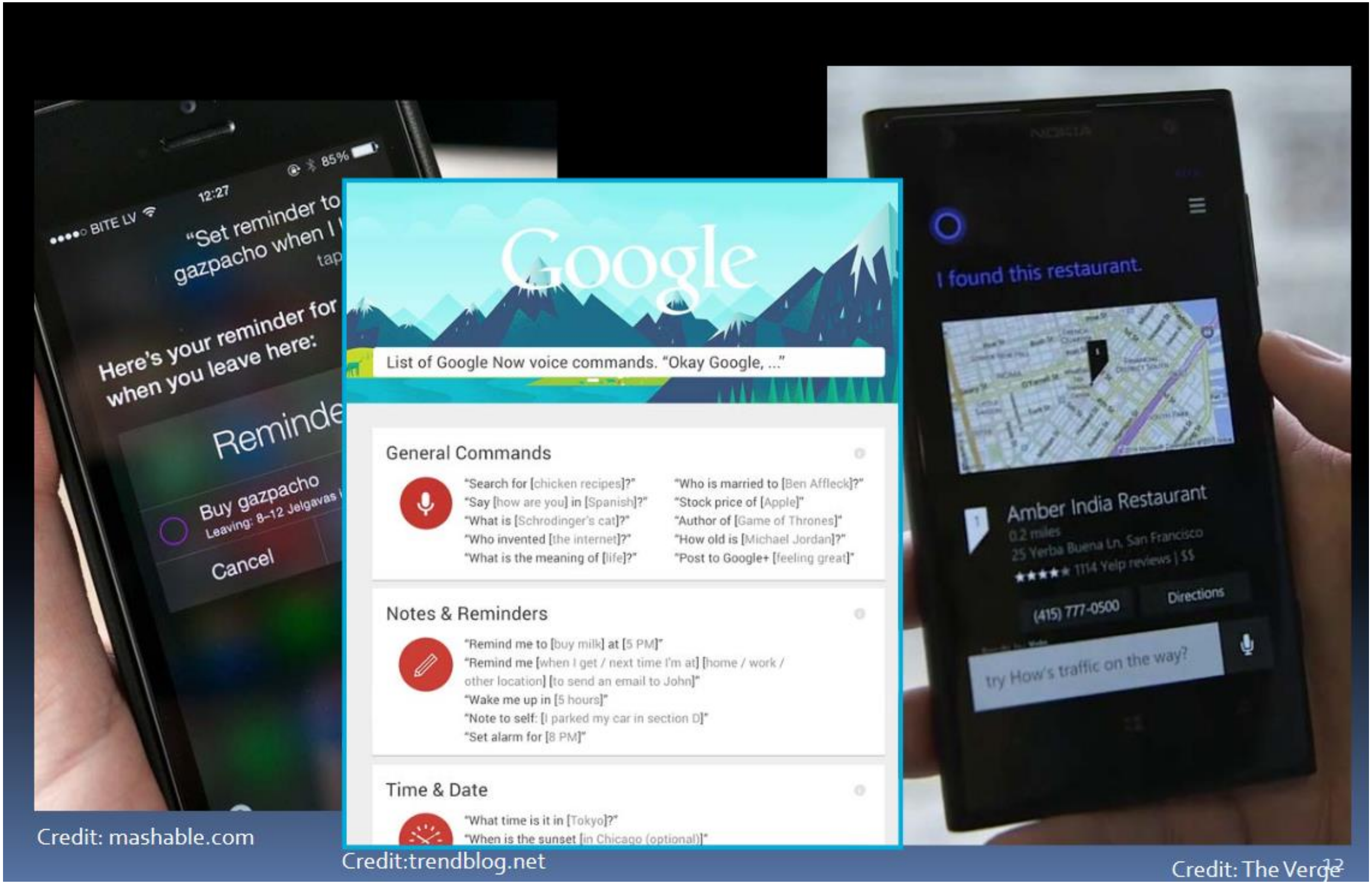
ML Successes: Chess and Go



Silver, et al. (2016) *Nature*
Deep Learning +
Monte Carlo Tree Search

Credit: Martin Mueller

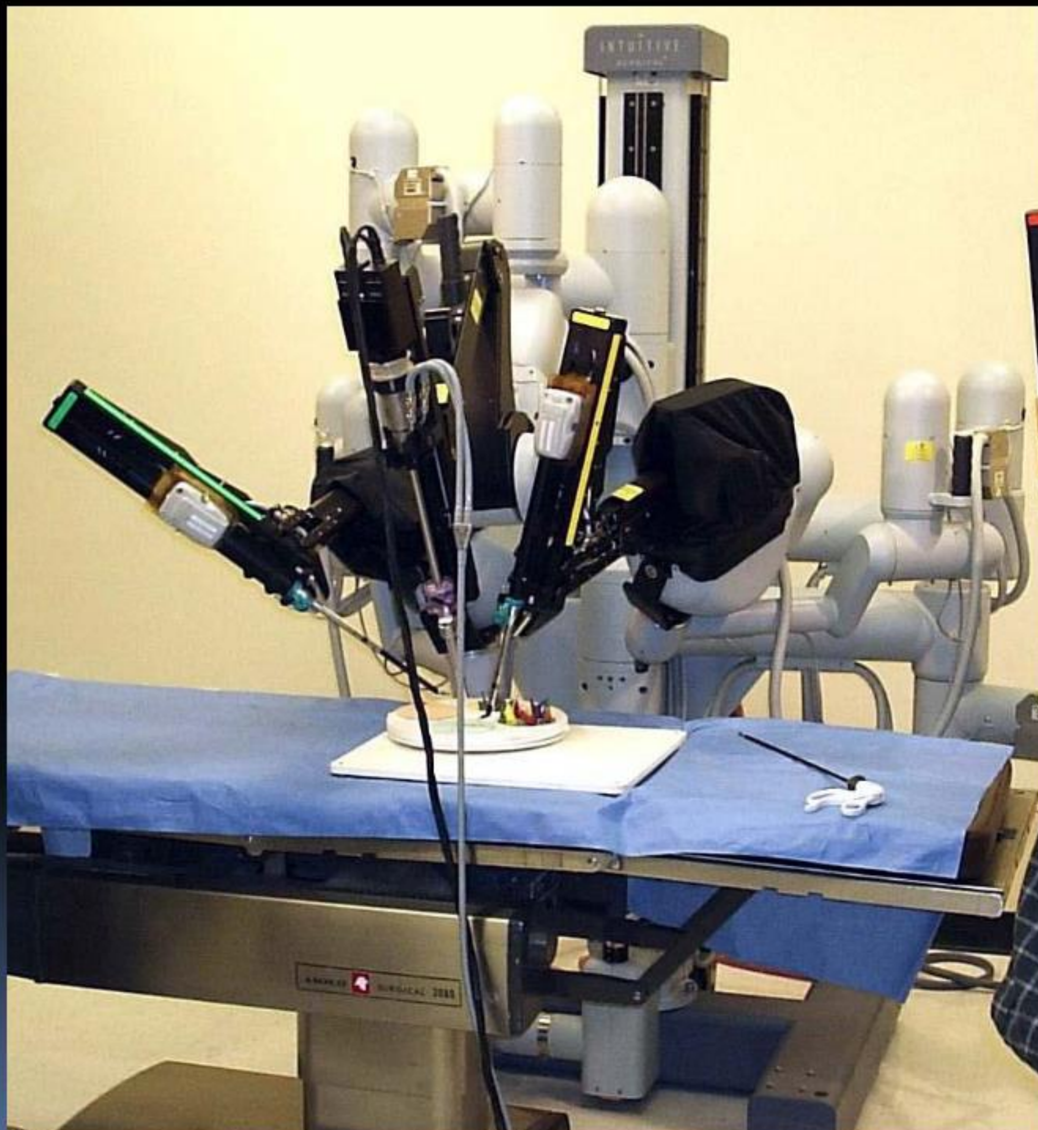
ML Successes: Personal Assistants



High-Stakes Applications: Self-Driving Cars

High-Stakes Applications: Automated Surgical Assistants

DaVinci



High-Stakes Applications:
AI Hedge Funds

WIRED

CADE METZ BUSINESS 01.25.16 7:00 AM

THE RISE OF THE ARTIFICIALLY INTELLIGENT HEDGE FUND

High-Stakes Applications: Power Grid Control

CONTROLLING THE POWER GRID WITH ARTIFICIAL INTELLIGENCE

02.07.2015

Credit: EBM Netz AG

DARPA Exploring Ways to Protect Nation's Electrical Grid from Cyber Attack

Effort calls for creation of automated systems to restore power within seven days or less after attack

Credit: DARPA

High-Stakes Applications: Autonomous Weapons

Northroop Grumman X-47B

Credit: Wikipedia

Samsung SGR-1

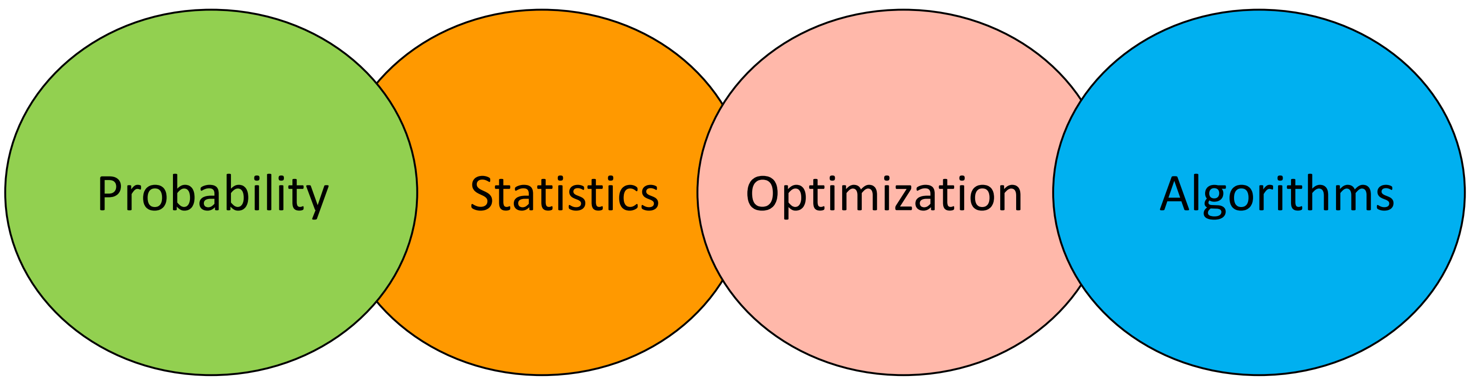
Credit: AFP/Getty Images

UK Brimstone Anti-Armor Weapon

Credit: Duch.seb - Own work, CC BY-SA 3.0

What is Machine Learning?

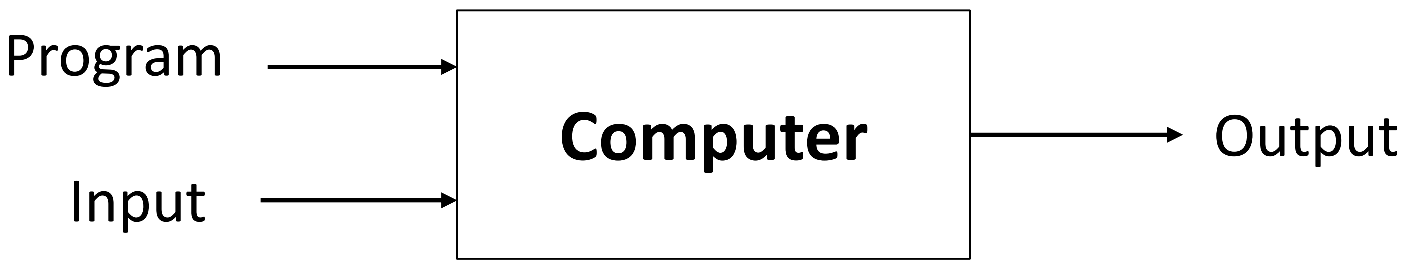
- **Machine learning is the branch of engineering that develops technology for automated inference**
  - It combines



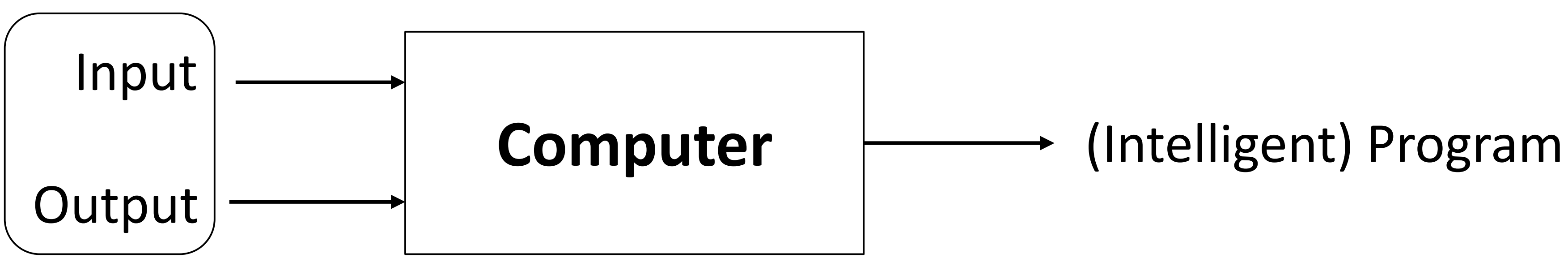
What is Machine Learning?

- **Machine learning = Automating Automation**

Traditional Programming



Machine Learning



Training data

Learning Paradigms

- **Supervised Learning –** main focus of this course

- **Semi-Supervised Learning**
- **Active Learning**
- **Reinforcement Learning**

Supervised Learning

Learning a Classifier

( , male)

Training Data

$\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$

X

Learning Algorithm

?

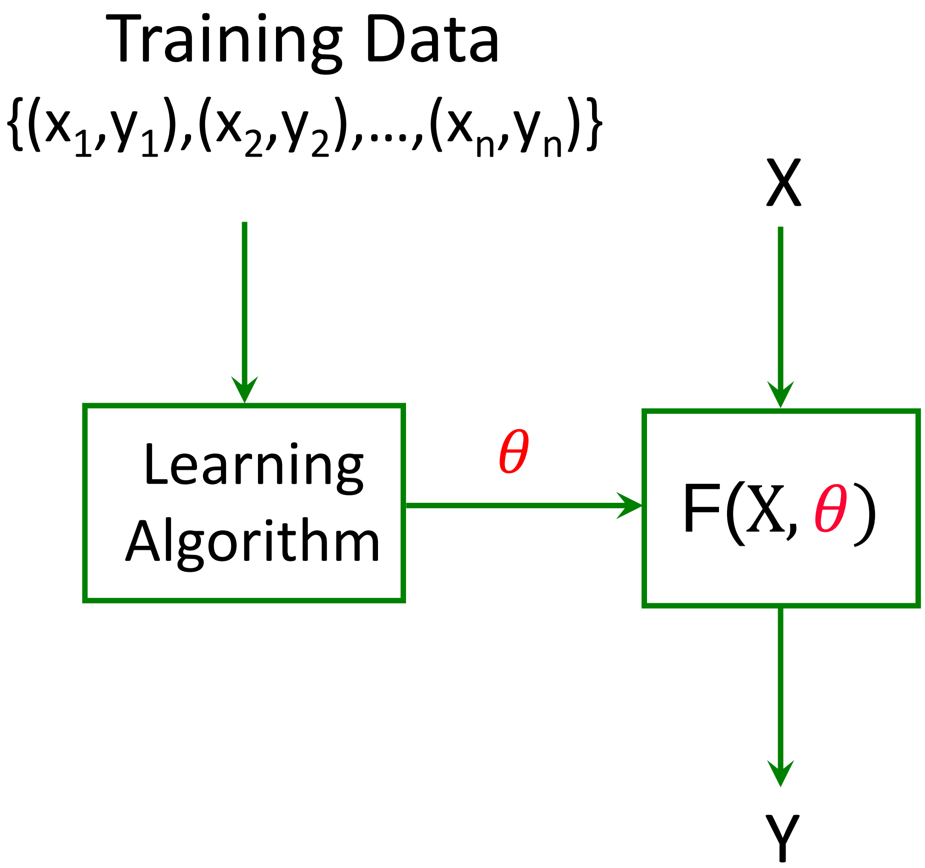
Y

<u>Example problem:</u>

X - image of a face

$Y \in$ {male, female}

Learning a Classifier

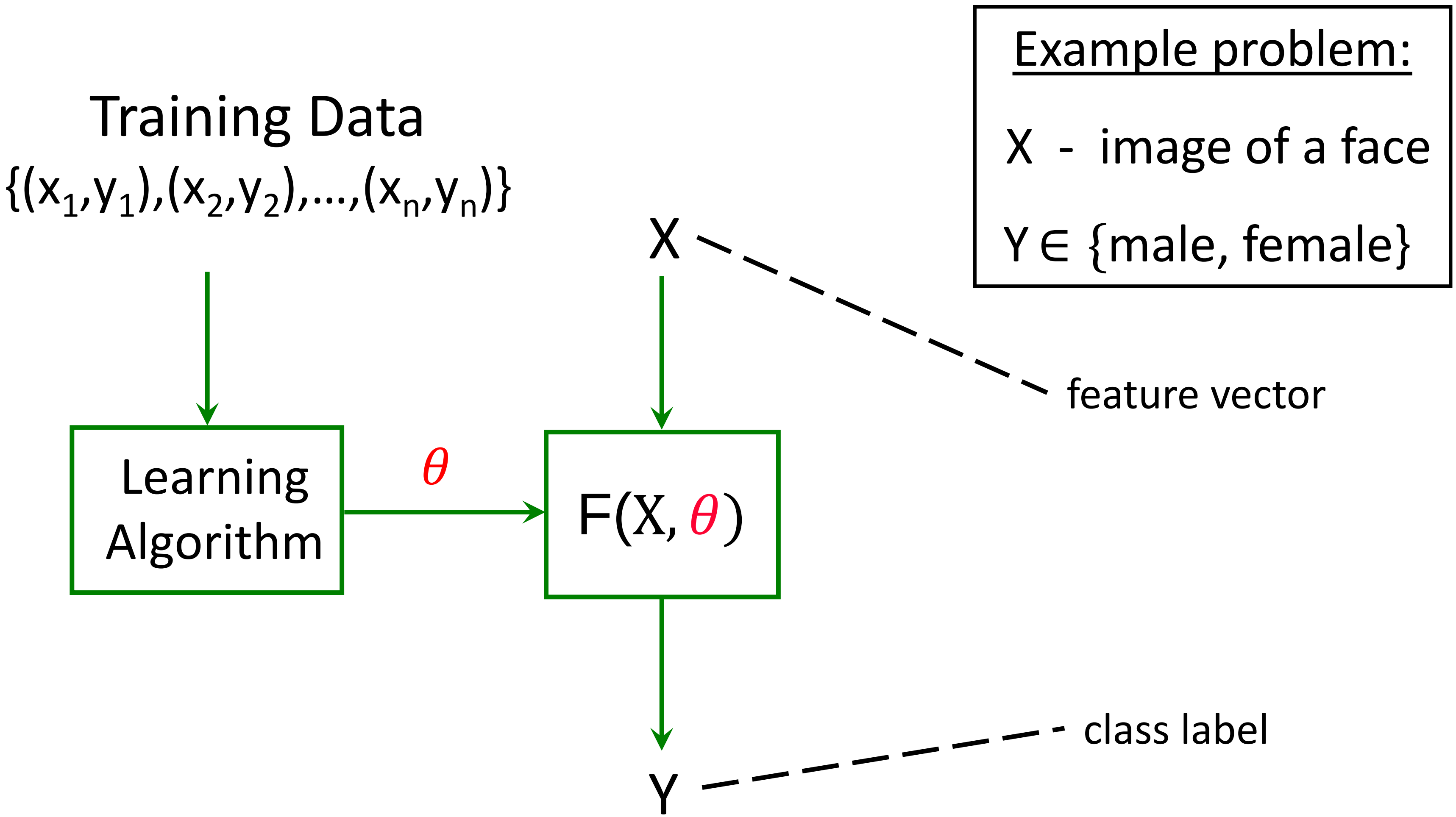


<u>Example problem:</u>

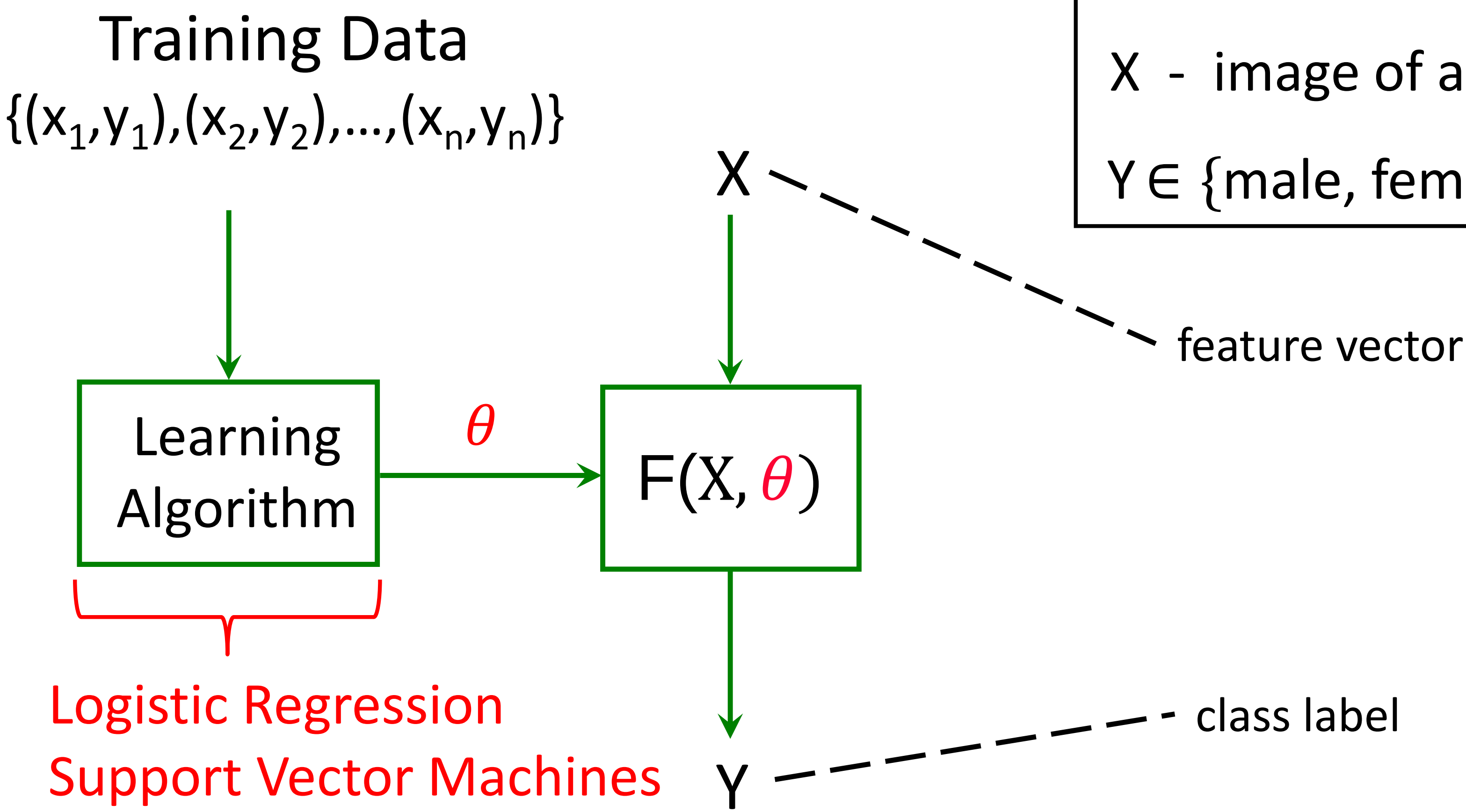
X - image of a face

$Y \in$ {male, female}

Learning for Simple Outputs



Learning for Simple Outputs



Example problem:

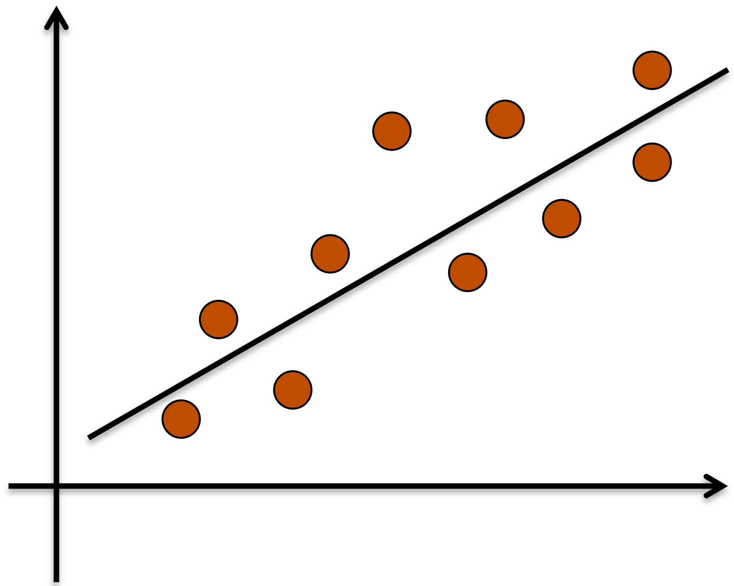
X - image of a face

Y ∈ {male, female}

Logistic Regression
Support Vector Machines
K Nearest Neighbor
Decision Trees
Neural Networks

Regression

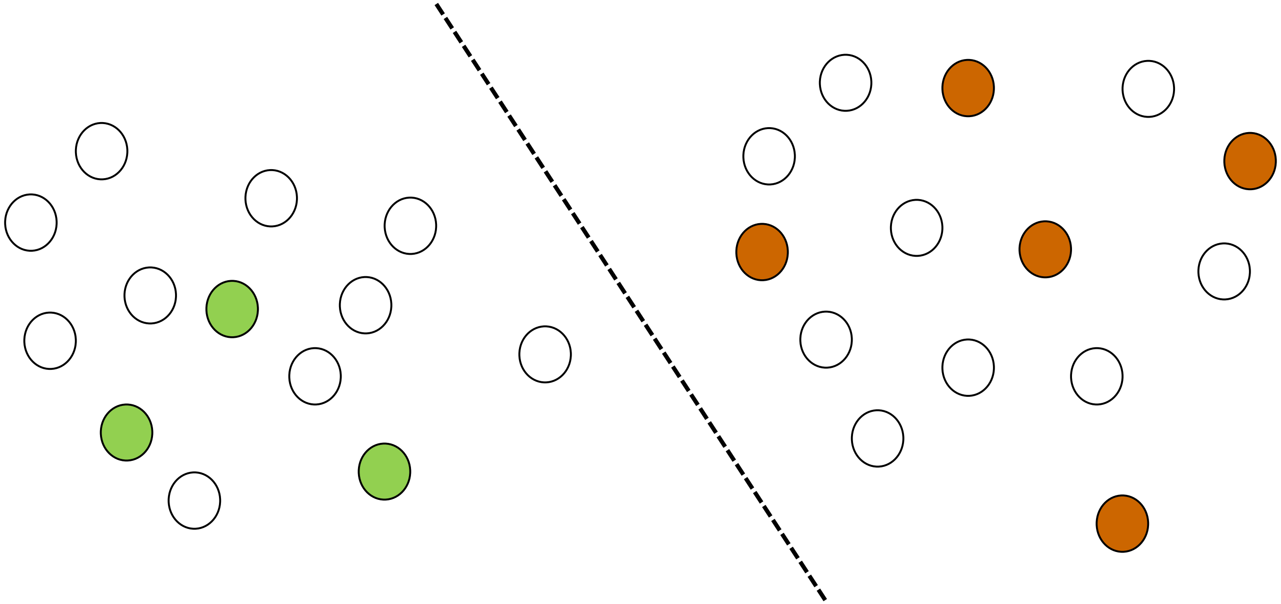
- **Setting:** output y is a continuous value instead of a discrete value
  - Stock market price as a function of financial specs



Semi-Supervised Learning

Semi-Supervised Learning

- **Setting:** small amount of labeled data and large amount of unlabeled data



- find a classifier that separates the labeled points and separates the unlabeled points "well"

Semi-Supervised Learning

- **Co-Training Style Algorithms**
  - Leverage diversity in the learners to learn from each other
  - Diversity comes from multiple (redundant) views of the input – In webpage classification, one view is the “words” on the page and another view is the “links” that point to that page
  - If only one view, employ learners with different hypothesis spaces to achieve diversity

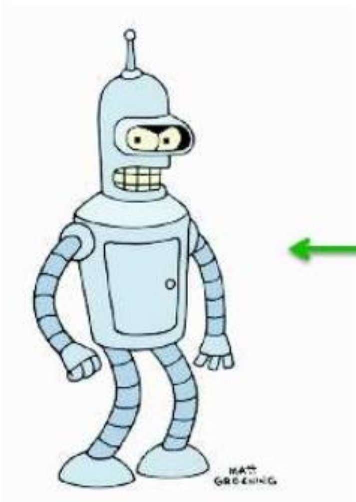
Active Learning

(Passive) Supervised Learning

raw unlabeled data

$x_1, x_2, x_3, \ldots$

random sample



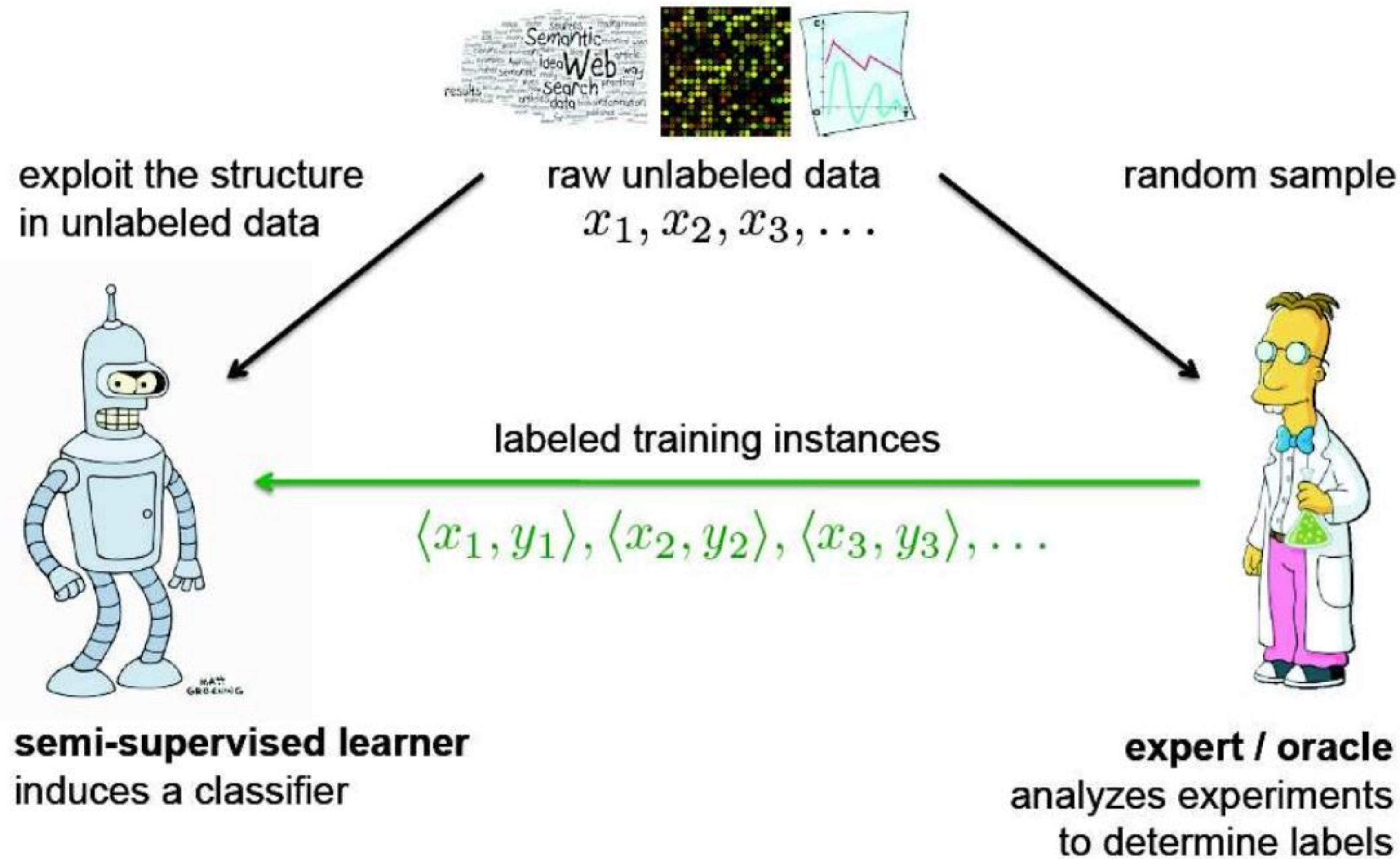
labeled training instances

$\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \langle x_3, y_3 \rangle, \ldots$

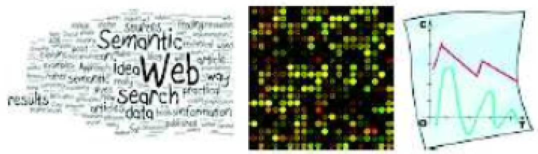
supervised learner
induces a classifier

expert / oracle
analyzes experiments
to determine labels

Semi-Supervised Learning

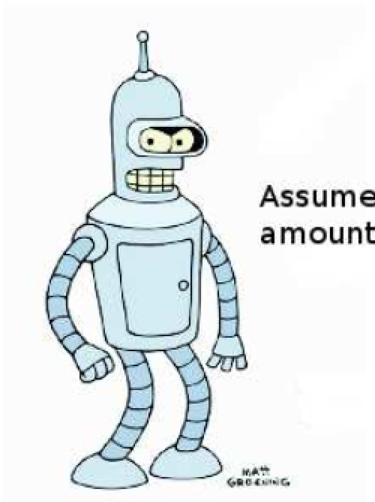


Active Learning



raw unlabeled data

$$x_1, x_2, x_3, \ldots$$

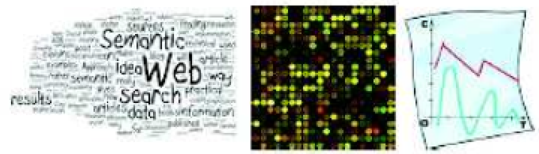


Assumes some small
amount of initial labeled training data

active learner
induces a classifier

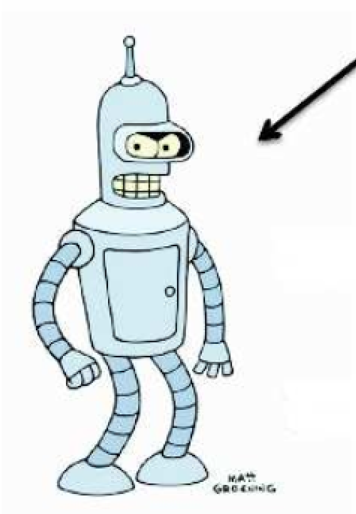
expert / oracle
analyzes experiments
to determine labels

Active Learning



raw unlabeled data
$x_1, x_2, x_3, \ldots$

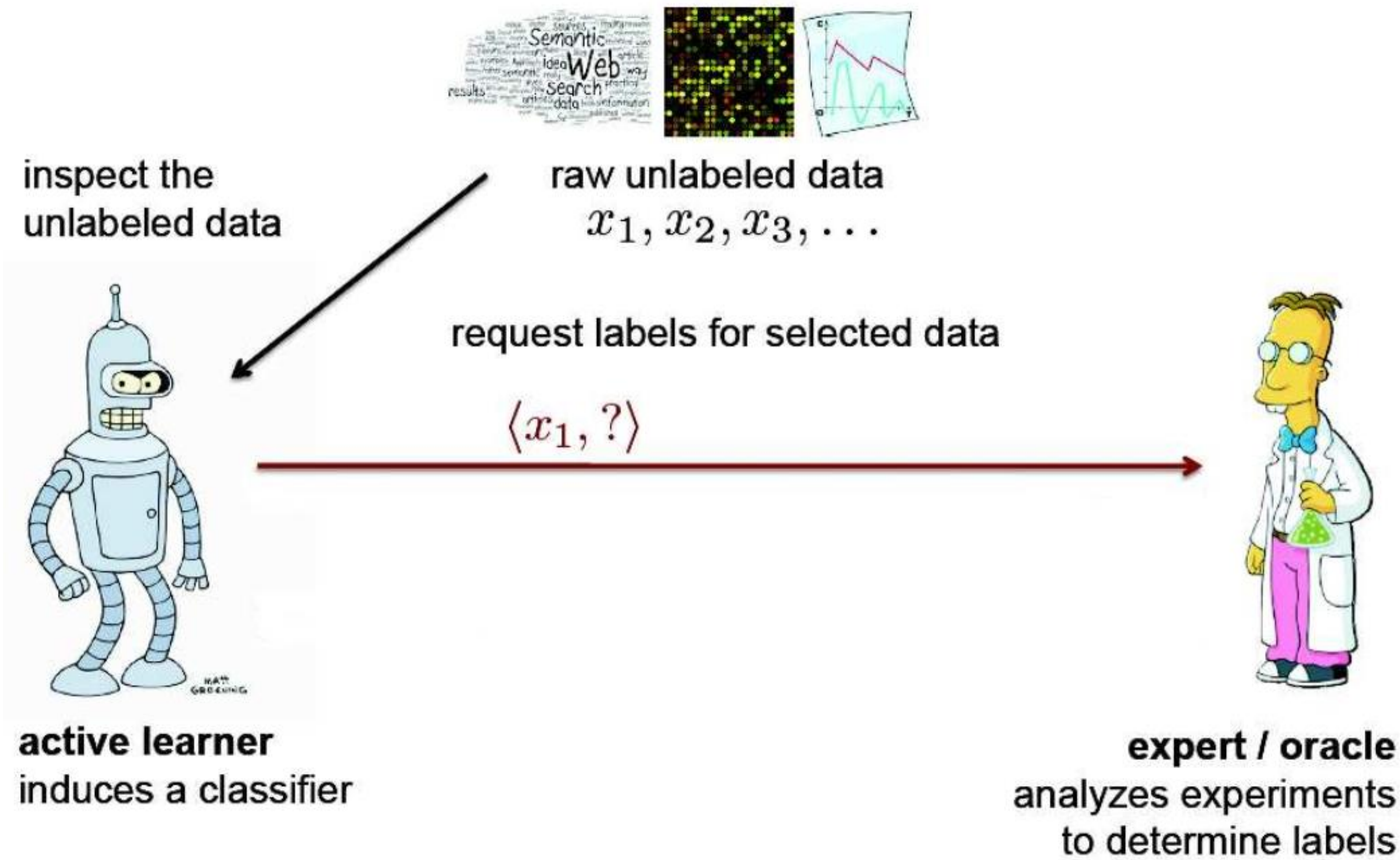
inspect the
unlabeled data



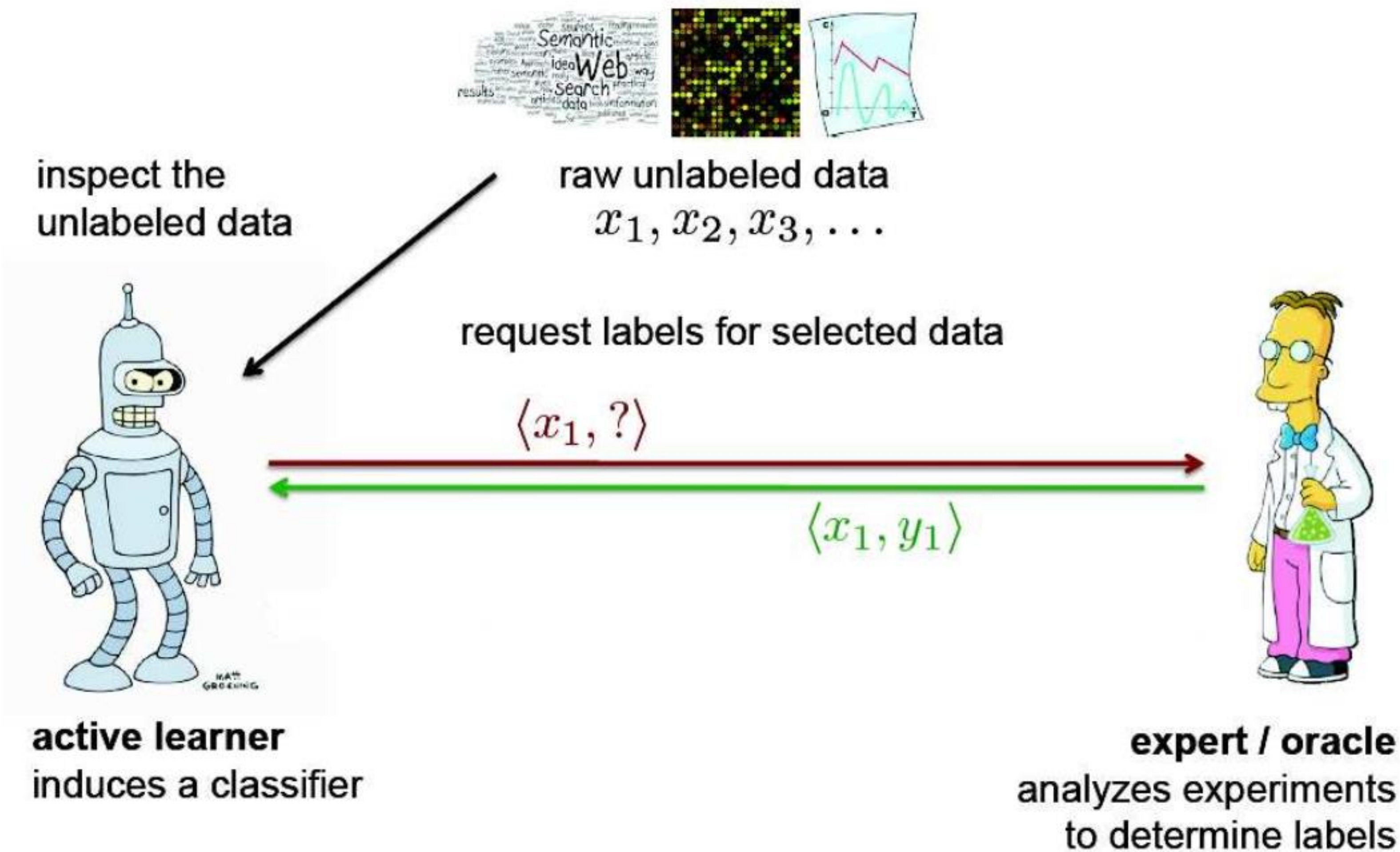
active learner
induces a classifier

expert / oracle
analyzes experiments
to determine labels

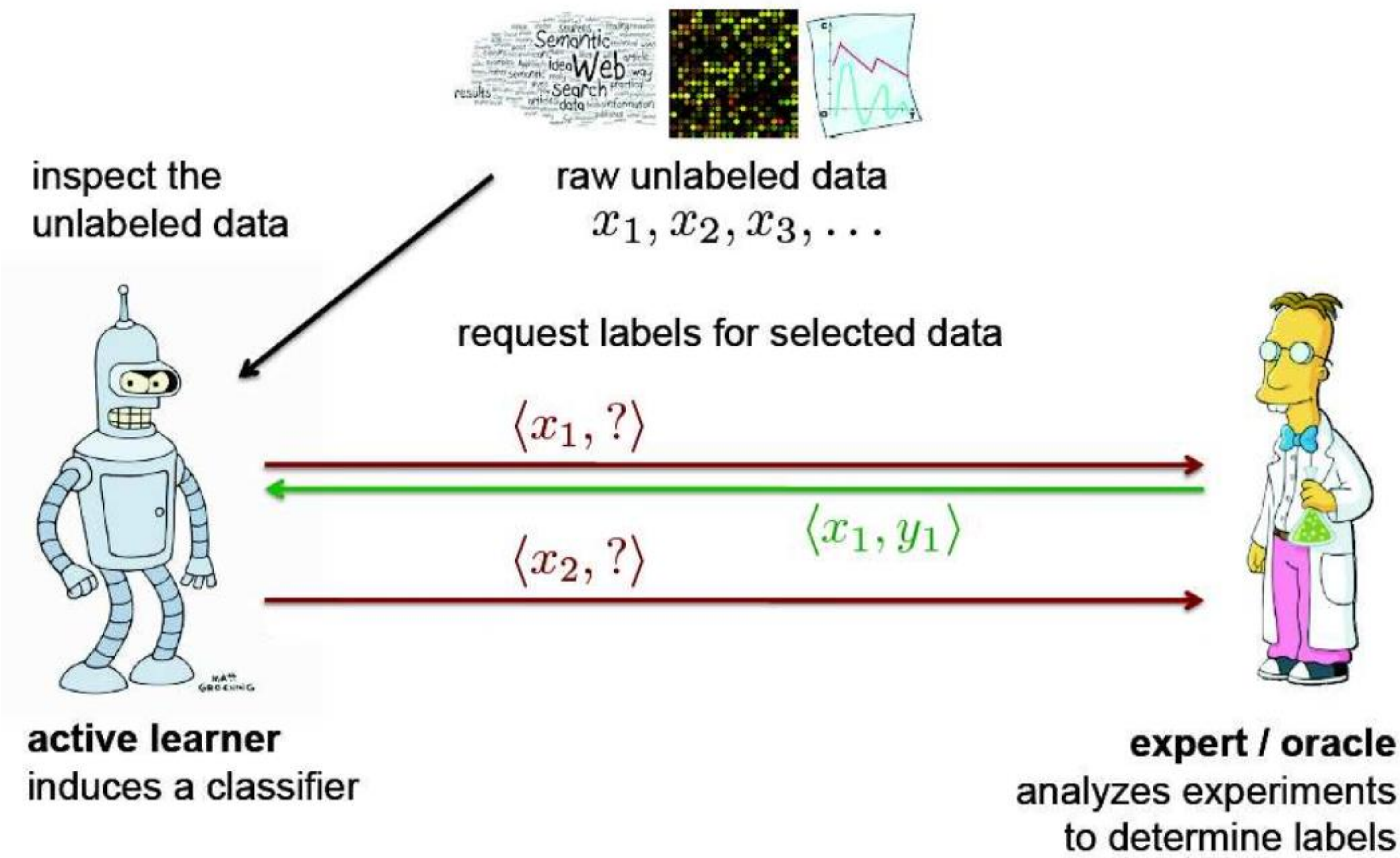
Active Learning



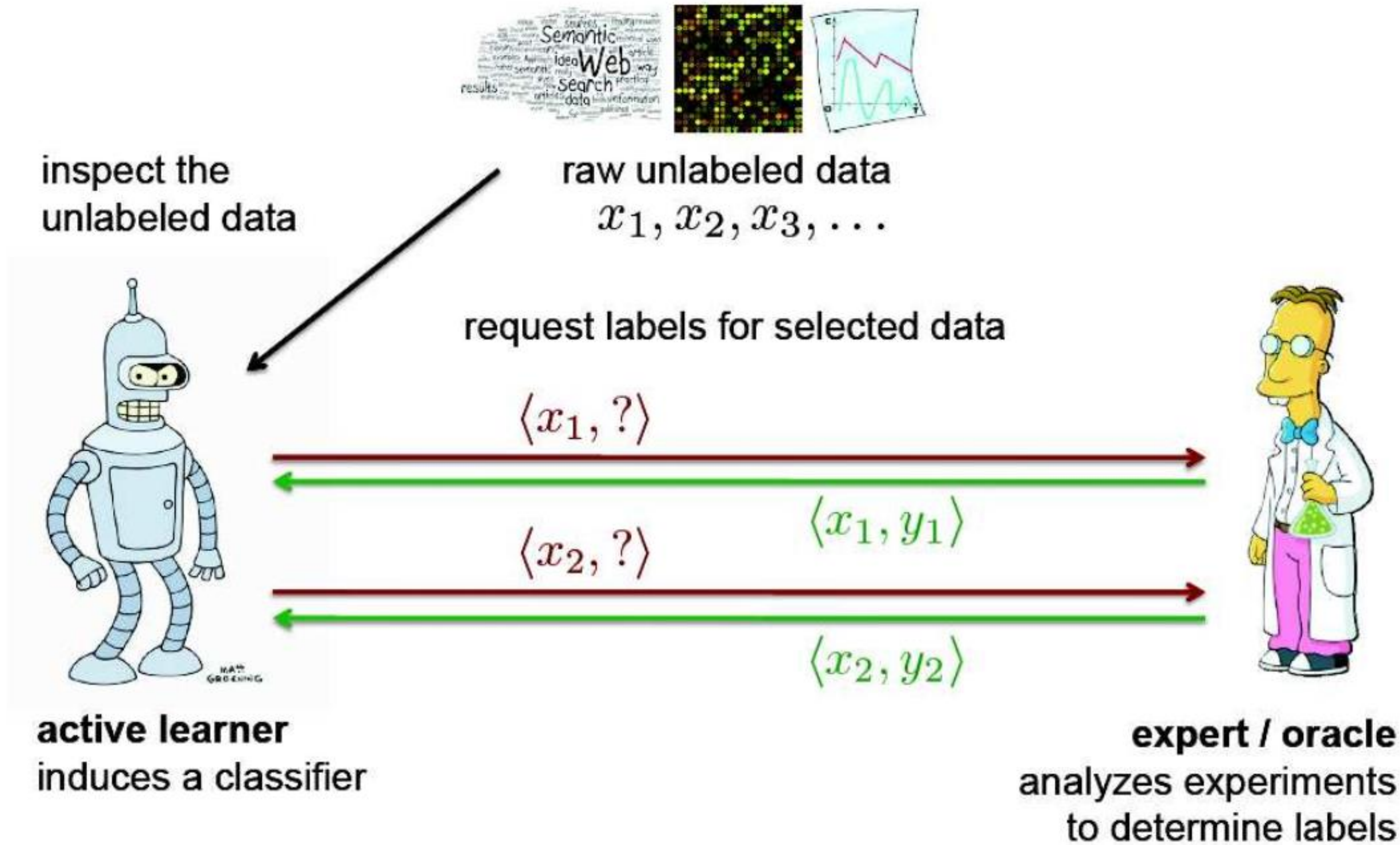
Active Learning



Active Learning



Active Learning

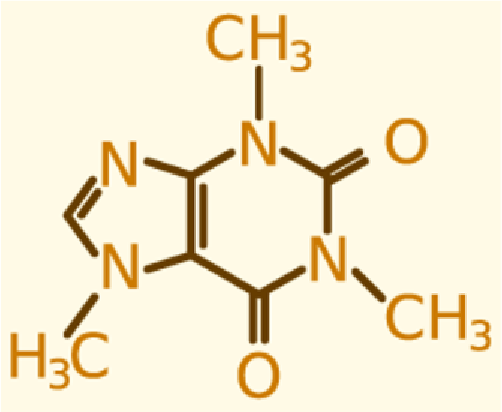


Motivation

- **Why do we need active learning?**
  - Supervised learning can solve all our problems, right?
  - Yes, if we have enough labeled data (input-output pairs)
  - But Labeling is expensive
  - We want to learn a highly-accurate function with few labeled examples
  - Intelligently select the examples for which we want to get labels for (unlabeled data is plentiful and cheap)

Active Learning Example: Drug Design

Goal: find compounds which bind to a particular target



Large collection of compounds, from:

- vendor catalogs
- corporate collections
- combinatorial chemistry

unlabeled point	≡	description of chemical compound
label	≡	*active* (binds to target) vs. *inactive*
getting a label	≡	chemistry experiment

Credit: Sanjoy Dasgupta and John Langford

Who uses Active Learning?

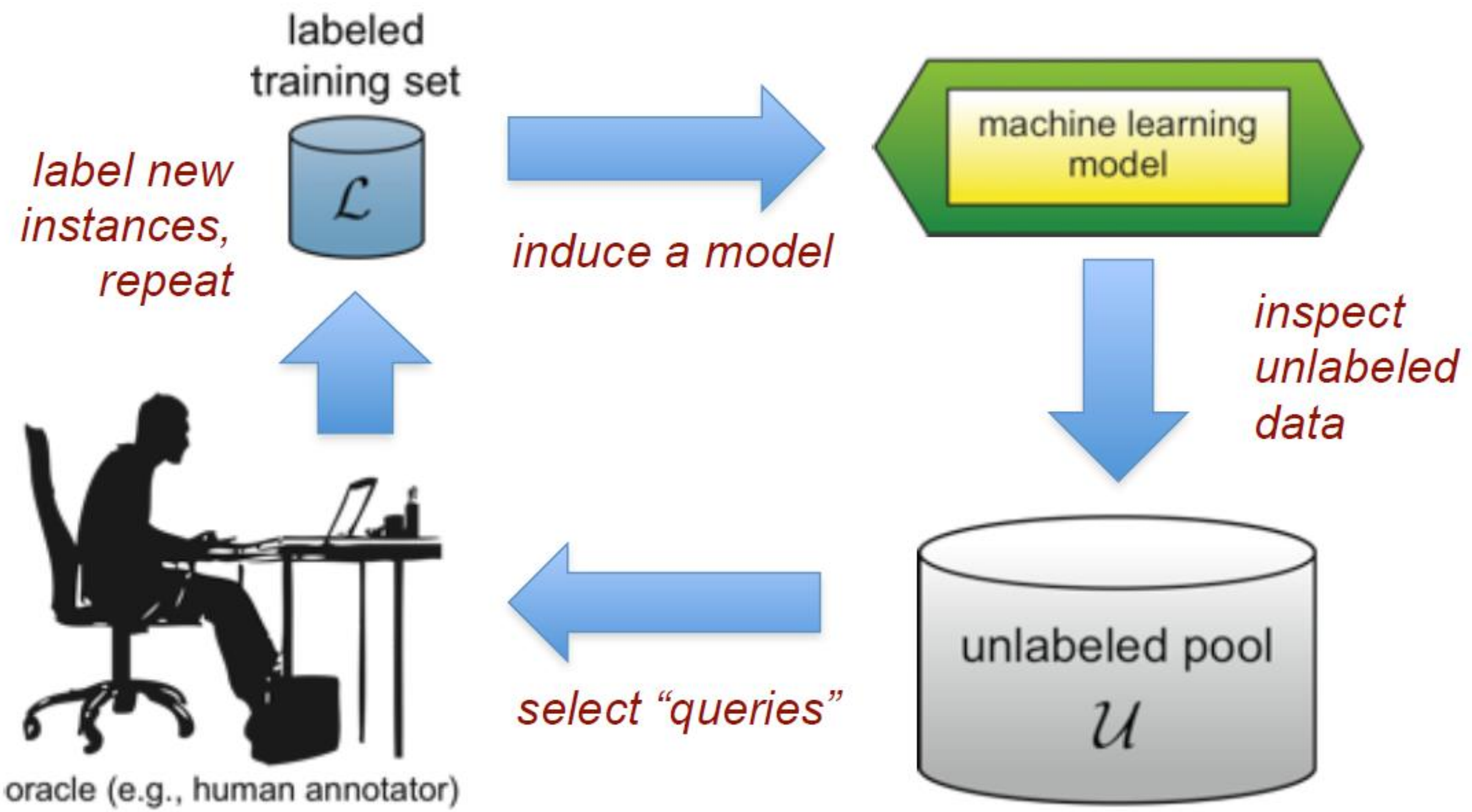
IBM: Sentiment analysis for blogs; Noisy relabeling
– *Prem Melville*

SIEMENS: Biomedical NLP & IR; Computer-aided diagnosis
– *Balaji Krishnapuram*

Microsoft: MS Outlook voicemail plug-in [Kapoor et al., IJCAI'07]; "A variety of prototypes that are in use throughout the company." – *Eric Horvitz*

Google: "While I can confirm that we're using active learning in earnest on many problem areas… I really can't provide any more details than that. Sorry to be so opaque!"
– *David Cohn*

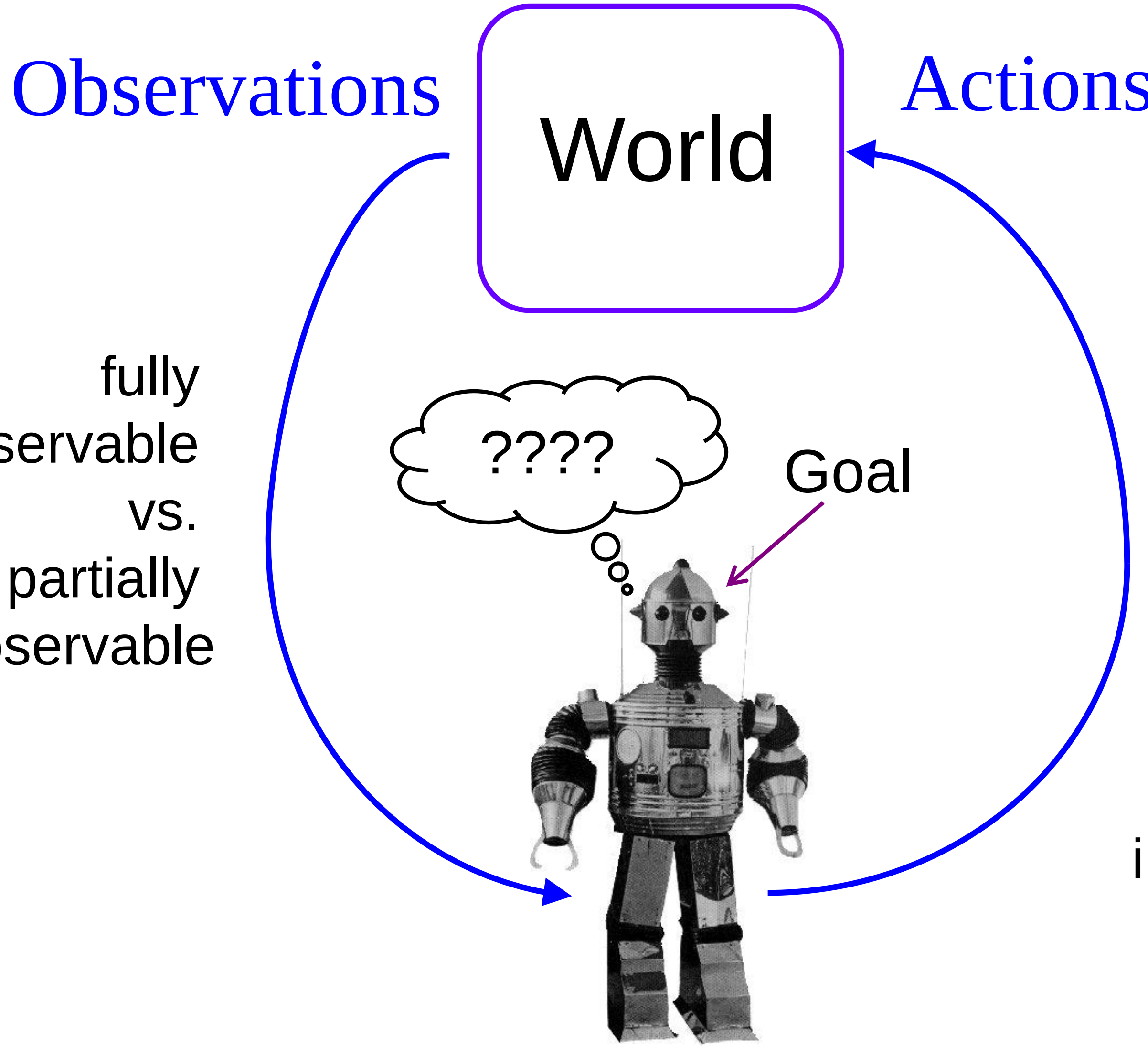
Pool based Active Learning



Credit: Burr Settles

Reinforcement Learning

Reinforcement Learning



fully
observable
vs.
partially
observable

sole source
of change
vs.
other sources

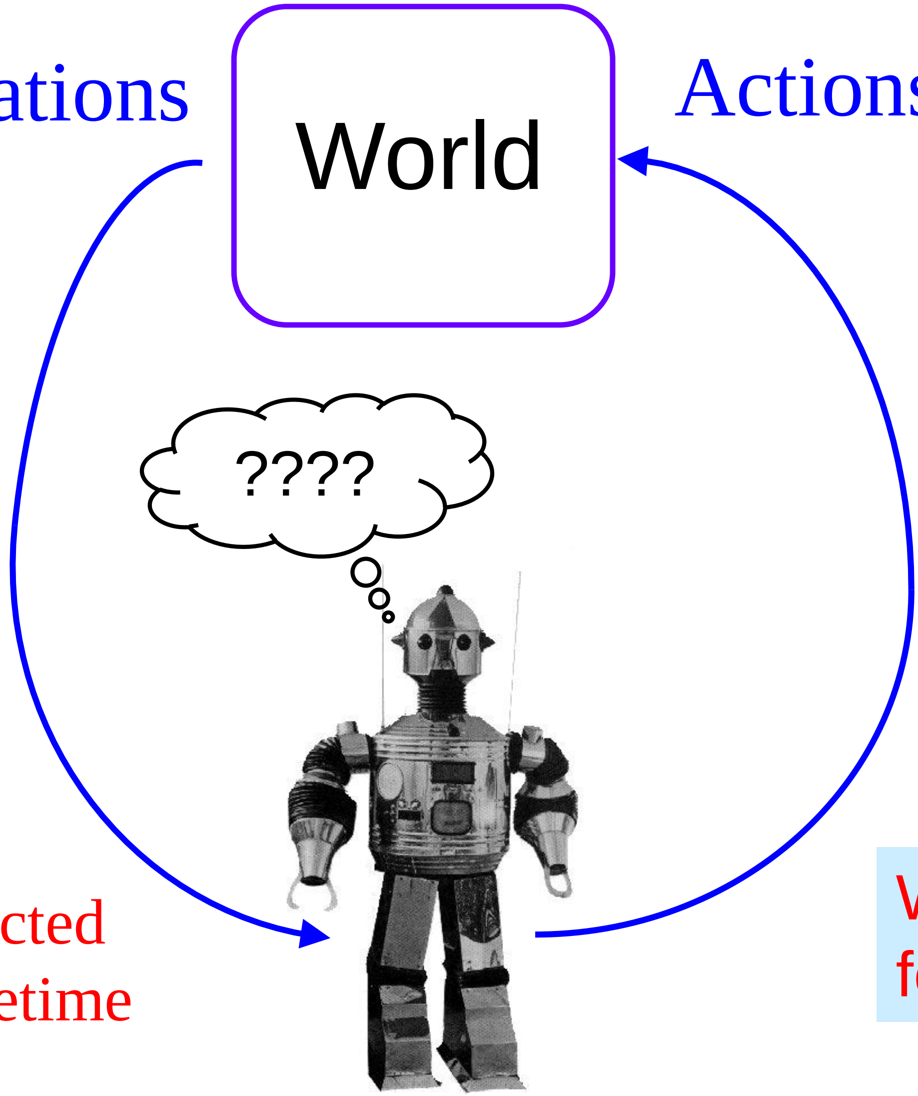
deterministic
vs.
stochastic

instantaneous
vs.
durative

Stochastic/Probabilistic Planning: Markov Decision Process (MDP) Model

Observations

Actions



fully
observable

sole source
of change

stochastic

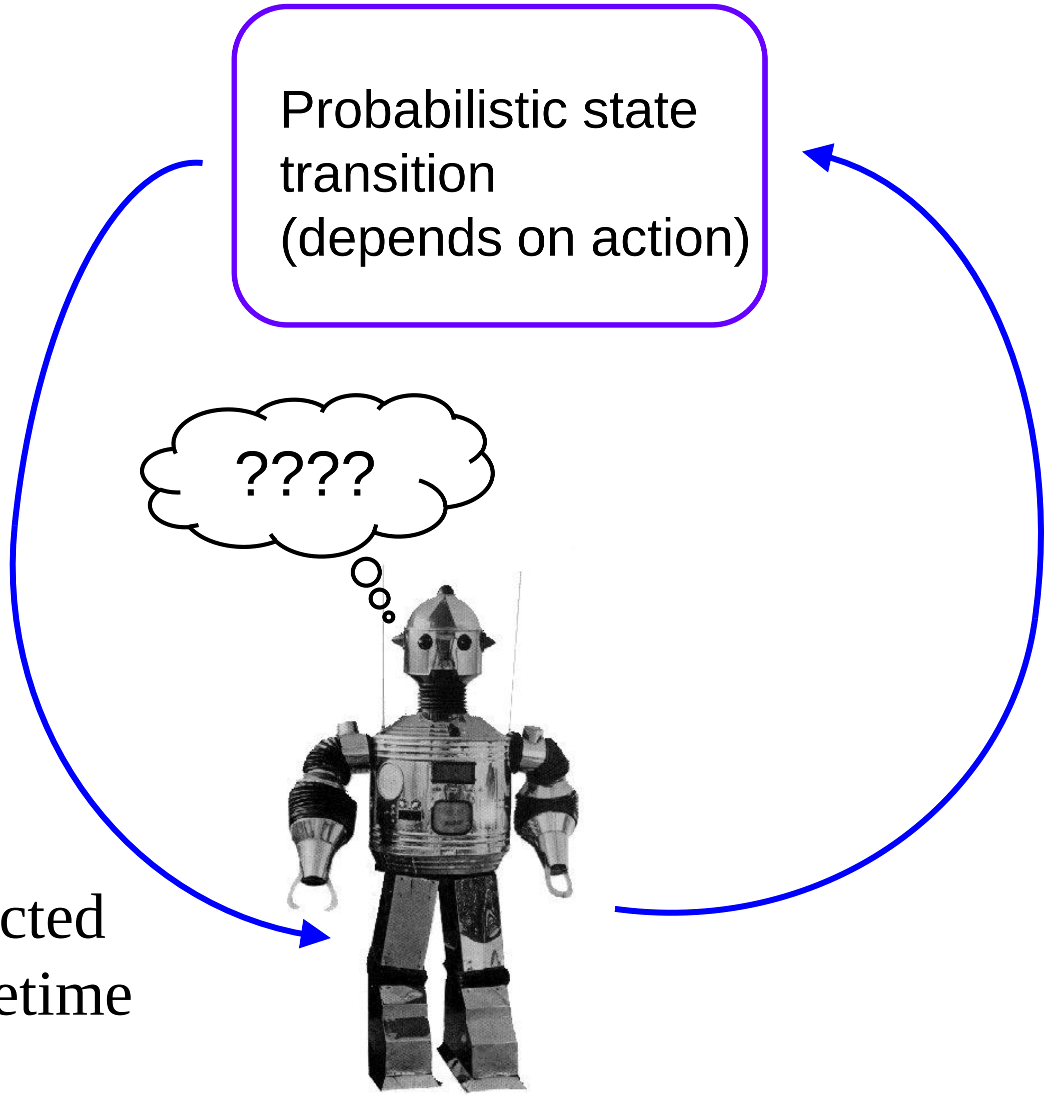
instantaneous

Goal

maximize expected
reward over lifetime

We will primarily
focus on MDPs

Stochastic/Probabilistic Planning: Markov Decision Process (MDP) Model



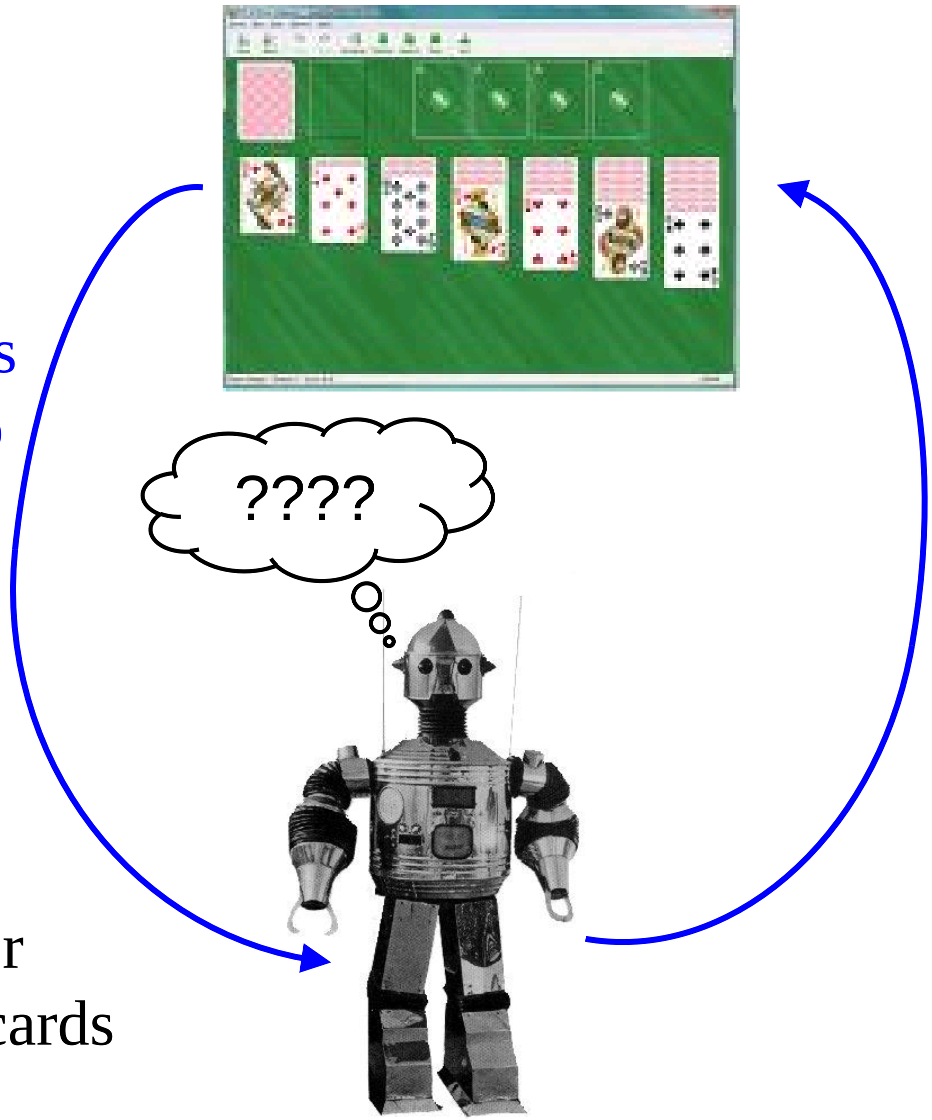
World State

Action from finite set

Goal

maximize expected reward over lifetime

Example MDP



State describes all visible info about cards

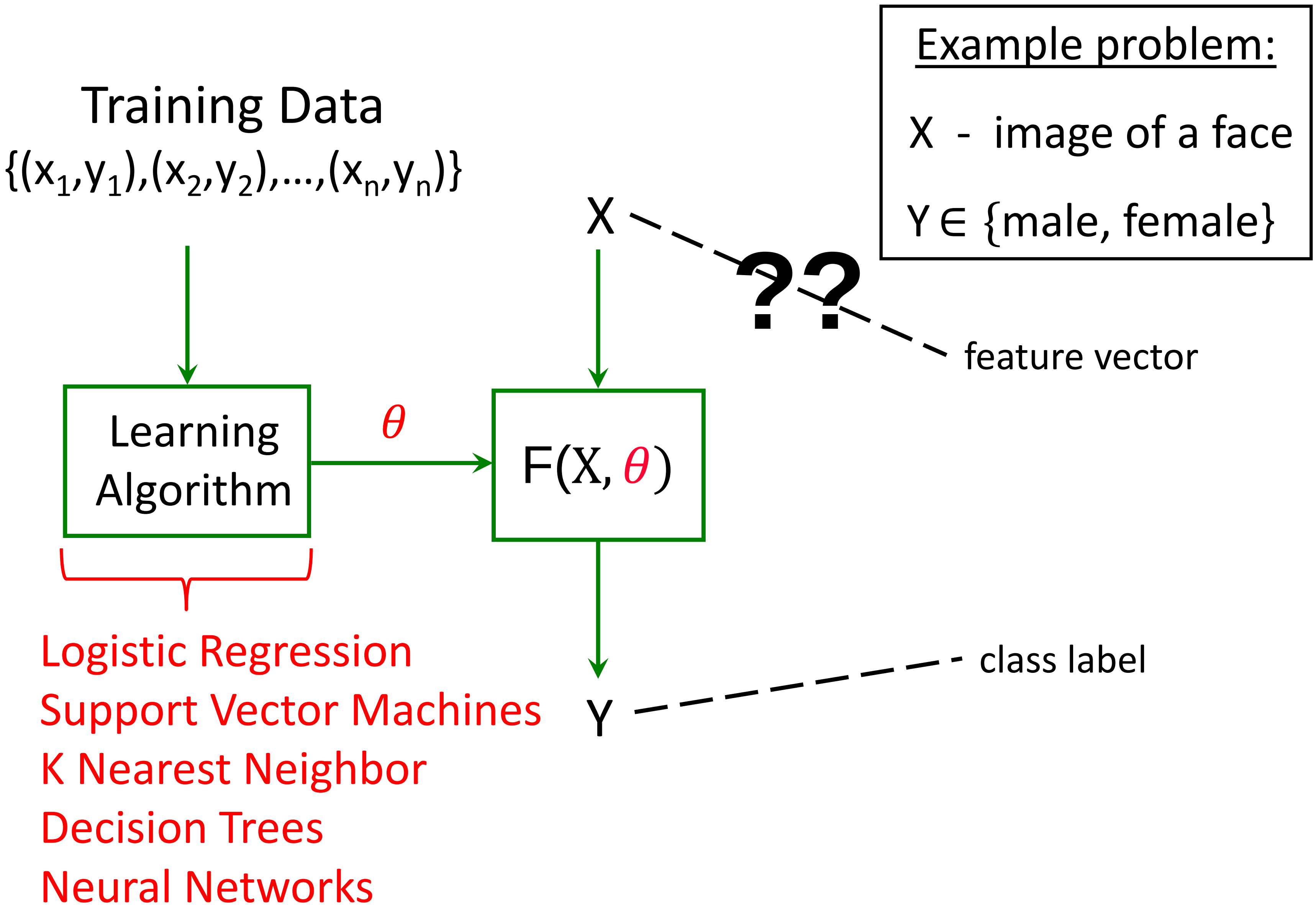
Action are the different legal card movements

Goal

win the game or play max # of cards

Input Representation
and
Abstract Machine Learning Algorithm

Learning for Simple Outputs



Input examples: Representation

- In ML, our input examples (emails, text documents, images) are often represented as real-valued vectors: $x \in R^d$
  - each co-ordinate of x is called a feature

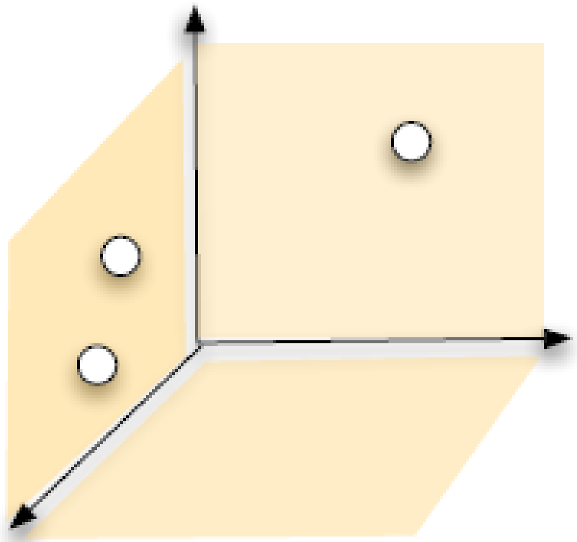
- Some examples
  - Bag-of-words representation of text
  - Histograms of colors in image
  - Sound frequency histogram

Input examples: Representation

- Bag-of-words model
  - sentences to points

1. To be, or not to be,
2. To be a woman,
3. To not be a man

To	be	or	not	woman	a	man
2	2	1	1	0	0	0
1	1	0	0	1	1	0
1	1	0	1	0	1	1

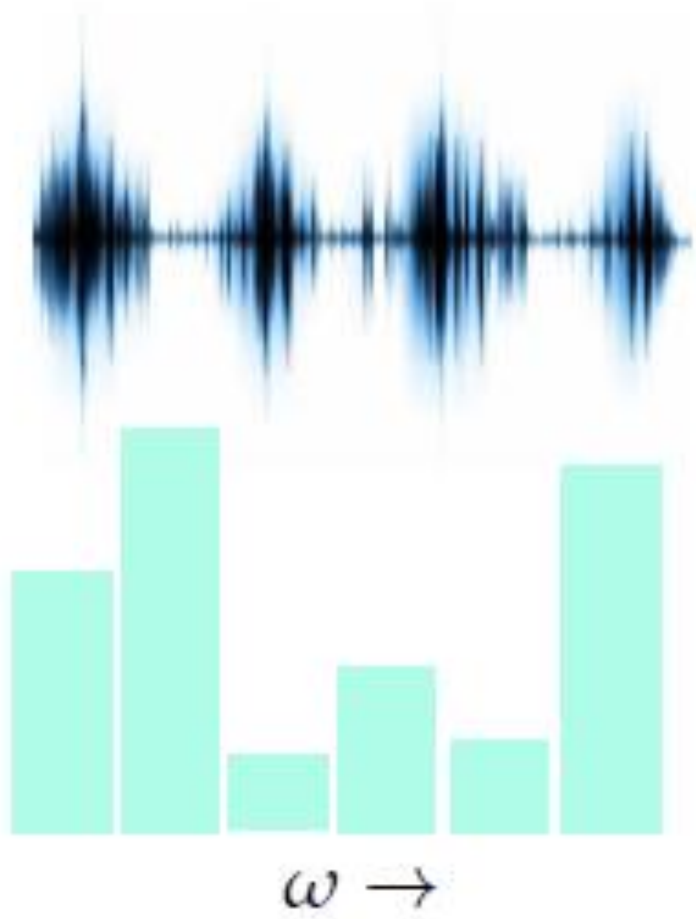


Input examples: Representation

- Histogram of colors in image

Input examples: Representation

- Sound frequency histogram



Overview of ML Algorithms

- There are lot of machine learning algorithms

- Every machine learning algorithm has three components
  - **Representation**
  - **Evaluation**
  - **Optimization**

Representation: Examples

- Linear hyper-planes
- Decision trees
- Sets of conjunctive / logical rules
- Graphical models (Bayes/Markov nets)
- Neural Networks
- …

Evaluation: Examples

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Cost / Utility
- Margin
- Entropy
- …

Optimization: Examples

- **Combinatorial Optimization**
  - greedy search, dynamic programming
- **Convex Optimization**
  - gradient descent, co-ordinate descent
- **Constrained Optimization**
  - linear programming, quadratic programming
- …

Machine Learned Programs: Errors

- **Approximation Error**
  - Error due to restricted hypothesis class (representation)
- **Estimation Error**
  - Error due to finite training samples
- **Optimization Error**
  - Error due to not finding a global optimum to the optimization problem

Learning Classifiers
via
Perceptron Algorithm

Formal setting – Classification

- **Instances** $\mathbf{x} \in \mathcal{X}$
  - emails
- **Labels** $y \in \mathcal{Y} = \{-1 \; ; \; 1\}$
  - Spam vs. non-spam
- **Prediction rule** $f(\mathbf{x}) = \hat{y}$
  - Linear prediction rule
- **Loss** $\ell(\hat{y}, y) \in \mathbb{R}_+$
  - No. of mistakes

Predictions

- Continuous predictions : $f : \mathcal{X} \to \mathbb{R}$
  - Label $\mathrm{sign}(f(\mathbf{x}))$
  - Confidence $|f(\mathbf{x})|$

- Linear Classifiers
  - Prediction :

$$\begin{aligned} \hat{y} &= \mathrm{sign}(f(\mathbf{x})) \\ &= \arg\max_{y \in \mathcal{Y}} \mathbf{w} \cdot \Phi(\mathbf{x}, y) \\ &= \mathrm{sign}(\mathbf{w} \cdot \mathbf{x}) \end{aligned}$$

$$|f(\mathbf{x})| = |\mathbf{w} \cdot \mathbf{x}|$$

Loss Functions

- **Natural Loss:**
  - Zero-One loss

$$\ell(\widehat{y}, y) = \begin{cases} 0 & y = \widehat{y} \\ 1 & y \neq \widehat{y} \end{cases}$$

Online Framework

- Initialize Classifier $f_1(\mathbf{x})$
- Algorithm works in rounds $t = 1 \ldots T \ldots$
- On round t the online algorithm :
  - Receives an input instance $\mathbf{x}_t$
  - Outputs a prediction $f_t(\mathbf{x}_t) = \widehat{y}_t$
  - Receives a feedback label y_t
  - Computes loss $\ell(\widehat{y}_t, y_t)$
  - Updates the prediction rule $f_t \rightarrow f_{t+1}$
- Goal :
  - Suffer small cumulative loss $\sum_t \ell(\widehat{y}_t, y_t)$

Why Online Learning?

- Fast
- Memory efficient - process one example at a time
- Simple to implement
- Formal guarantees – Mistake bounds
- Online to Batch conversions
- No statistical assumptions
- Adaptive

Update Rules

- Online algorithms are based on an update rule which defines f_{t+1} from f_t (and possibly other information)
- **Linear Classifiers** : find $\mathbf{w}_{t+1}$ from $\mathbf{w}_t$ based on the input $(\mathbf{x}_t, y_t)$
- **Perceptron algorithm employs a specific update rule**

Design Principle of Online Learning Algorithms

- If the learner suffers non-zero loss at any round, then we want to balance two goals:
  - **Corrective:** Change weights so that we don’t make this error again
  - **Conservative:** Don’t change the weights too much

The Perceptron Algorithm ($\eta = 1$)

- If No-Mistake $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) > 0$
  - Do nothing $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
- If Mistake $y_t(\mathbf{w}_t \cdot \mathbf{x}_t) \leq 0$
  - Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_t \mathbf{x}_t$

The Perceptron Algorithm ($\eta = 1$)

- When mistake happens, what does the update do?
  - **If $\boldsymbol{y_t = 1}$:** $\boldsymbol{w_{t+1} = w_t + x_t}$
    - ✓ w_{t+1} moves "closer to" x_t OR
    - ✓ x_t moves towards the positive side of the decision boundary
  - **If $\boldsymbol{y_t = -1}$:** $\boldsymbol{w_{t+1} = w_t - x_t}$
    - ✓ w_{t+1} moves "away from" x_t OR
    - ✓ x_t moves towards the negative side of the decision boundary
- In both cases, we are moving towards the "correct solution"

The Perceptron Algorithm

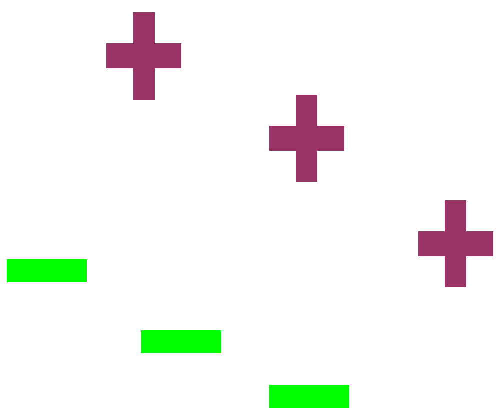
- **Suppose w_t makes a mistake on (x_t, y_t), and we update w_{t+1}as $w_{t+1} = w_t + y_t x_t$. Is it possible for w_{t+1} to also make a mistake on (x_t, y_t) ?**

The Perceptron Algorithm ($\eta = 1$)

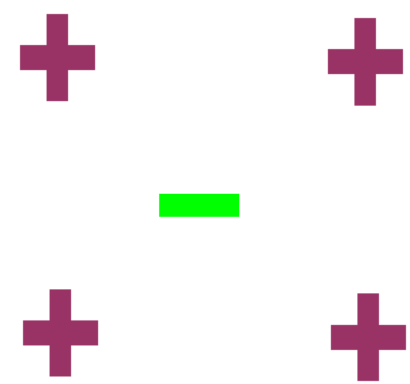
- **Suppose w_t makes a mistake on (x_t, y_t), and we update w_{t+1}as $w_{t+1} = w_t + y_t x_t$. Is it possible for w_{t+1} to also make a mistake on (x_t, y_t) ?**
  -
  - Yes, depends on the learning rate η

When does Perceptron converge?

- **Linear Separability**
  - There exists a hyper-plane (weight vector) separating the positive and negative points



Linearly separable



Not linearly separable

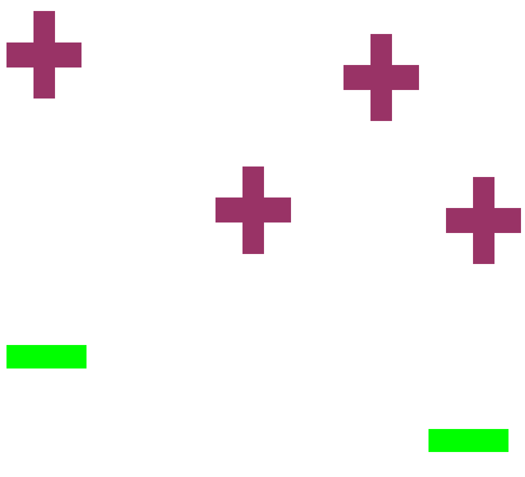
Measure of Separability

- **Margin**
  - For a weight vector $\boldsymbol{w}$, and training set $\boldsymbol{S}$, margin of w with respect to S is defined as follows:

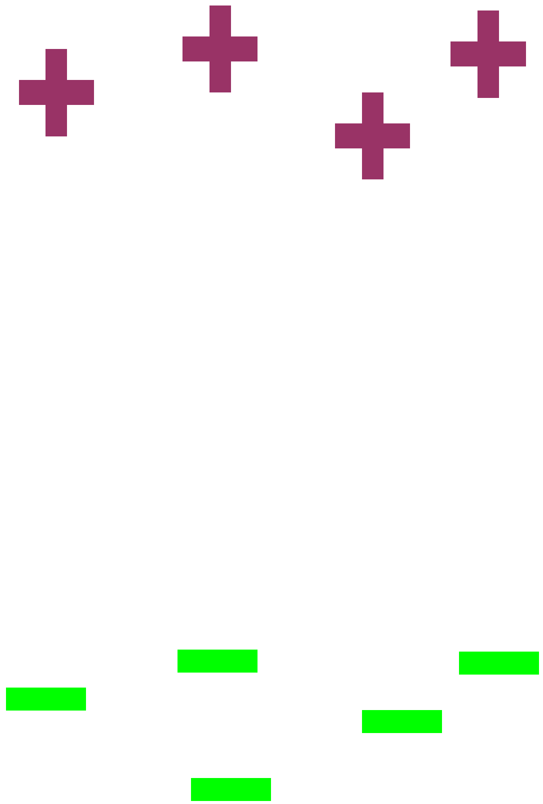
$$\boldsymbol{\gamma(w) = min_{(x,y)\in S}} \; y(w.x)$$

- The training data $\boldsymbol{S}$ is linearly separable if there exists *at least* one weight vector w for which the margin is positive, i.e., $\boldsymbol{\gamma(w) > 0}$.

Margin: Examples



Low margin data



High margin data

Perceptron: Convergence Result

- **Theorem:** If the training data is linearly separable with margin γ, and if $\|x_i\| \leq 1$ for all examples (x_i, y_i) in the training set, then perceptron makes $\leq \frac{1}{\gamma^2}$ mistakes.
  - Proof??

- Lower margin implies more mistakes

- May need more than one pass over the training data to get a classifier with no mistakes

What if data is not linearly separable?

- **Ideally, we want to find a linear separator that makes the minimum number of mistakes on the training data**
  - NP-Hard problem! (Minsky and Papert, 1969)
  - This result killed the neural networks research in 1970's

- **Perceptron still works**
  - there will be few mistakes close to the decision boundary
  - will never converge to a single w as we make more passes

Problems with Perceptron

- Doesn’t converge with inseparable data
- Weight updates may often be very “bold”
- Doesn’t optimize margin
- Sensitive to the order of examples

▲ **Voted and Averaged perceptron**

Voted Perceptron

- **Initialization:** $m = 1;\ w_1 = 0;\ c_m = 1$
- **Training Examples:** for $t = 1, 2, 3, \ldots$
  - If mistake, update weights
    - $w_{m+1} = w_m + y_t x_t$
    - $m = m + 1$
    - $c_m = 1$
  - Else
    - $c_m = c_m + 1$ // counting how long w_m survived
- **Output:** $(w_1, c_1), (w_2, c_2), (w_3, c_3), \ldots$

Voted Perceptron Classifier

$$\boldsymbol{f(x) = sign\left(\sum_{i=1}^{m} c_i \, sign(< w_i, x >)\right)}$$

- Any drawbacks of voted perceptron?

Voted Perceptron Classifier

$$f(x) = sign\left(\sum_{i=1}^{m} c_i \, sign(< w_i, x >)\right)$$

- Any drawbacks of voted perceptron?
- Yes, we have to store all the classifiers (in practice could be many)
- How can we solve this problem?

Averaged Perceptron

- Same algorithm as voted perceptron, but the classification rule is different

$$f_{average}(x) = sign\left(\sum_{i=1}^{m} (< c_i w_i, x >)\right)$$

$$f_{voted}(x) = sign\left(\sum_{i=1}^{m} c_i \, sign(< w_i, x >)\right)$$

Averaged vs. Voted Perceptron

- Simple Example: If $c_1 = c_2 = c_3 = 1$

$$\boldsymbol{f_{average}(x) = sign(\langle w_1 + w_2 + w_3, x \rangle)}$$

$$\boldsymbol{f_{voted}(x) = majority\ sign\ of\ \langle w_1, x \rangle, \langle w_2, x \rangle, \langle w_3, x \rangle}$$

Some Practical Tricks

- **Shuffling**
  - shuffling the training examples in each iteration
- **Variable learning rate**
  - decrease as learning progresses
  - follow some schedule
  - Set automatically by line search (converges faster)
  - See Leon Bottou's SGD website: http://leon.bottou.org/projects/sgd
- **Averaged Perceptron can be implemented very efficiently** (See Algorithm 7 in Hal's chapter)

Some Practical Tricks

- **Learning Curve**
  - Training iterations vs. number of mistakes
  - You want to see that the mistakes decrease as we increase the no. of iterations (curve goes down)
  - Very useful in debugging and seeing the behavior of online learning algorithms
- **Hyper-parameter Optimization**
  - Split the training data: sub-train + validation data
  - Tune hyper-parameters (e.g., no. of iterations) on the validation data
  - The learner should not look at the test data!

Multi-Class Classification: Setup

Suppose we have $(K > 2)$ classes.

K weight vectors: $w_1, w_2, \ldots, w_K \in R^d$

input instance $x \in R^d$

score(label r) $= w_r \cdot x$

Multi-Class Classification: Learning

Class r	$w_r \cdot x$
1	-1.08
2	1.66
3	0.37
4	-2.09

Prediction: output label (class) with highest score.

Learning:

$$w_{y^*} = w_{y^*} + x$$

$$w_{\hat{y}} = w_{\hat{y}} - x$$

Regression Learning: Setup

Regression Learning:

y is continuous value.

Prediction Rule: $F(x) = w \cdot x$

Widrow-Hoff Algorithm:

- Initialize $w_1 = 0$

for $t = 1$ to T do

- get $x_t \in R^d$
- predict $\hat{y}_t = w_t \cdot x_t$
- observe y^*_t
- Incur loss of $(\hat{y}_t - y^*_t)^2$
- $w_{t+1} = w_t - \eta (w_t \cdot x_t - y^*_t) x_t$

end