Caltech

Machine Learning & Data Mining

CS/CNS/EE 155

Lecture 17:

Survey of Advanced Topics

What We Covered

Basic Supervised Learning

- Training Data: $S = \{(x_i, y_i)\}_{i=1}^{N}$ $\quad x \in R^D$ $\quad y \in \{-1, +1\}$

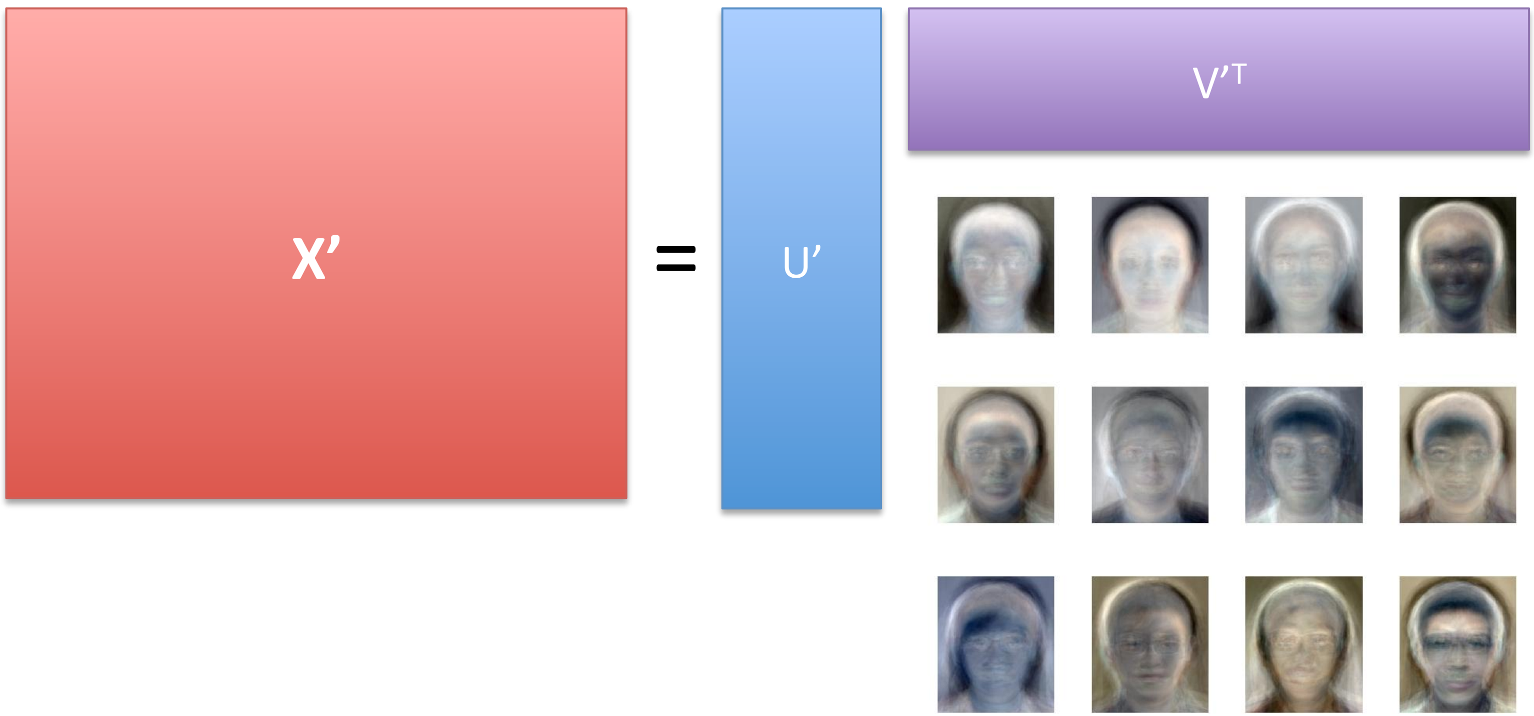
- Model Class: $f(x \mid w, b) = w^T x - b$ **Linear Models**

- Loss Function: $L(a, b) = (a - b)^2$ **Squared Loss**

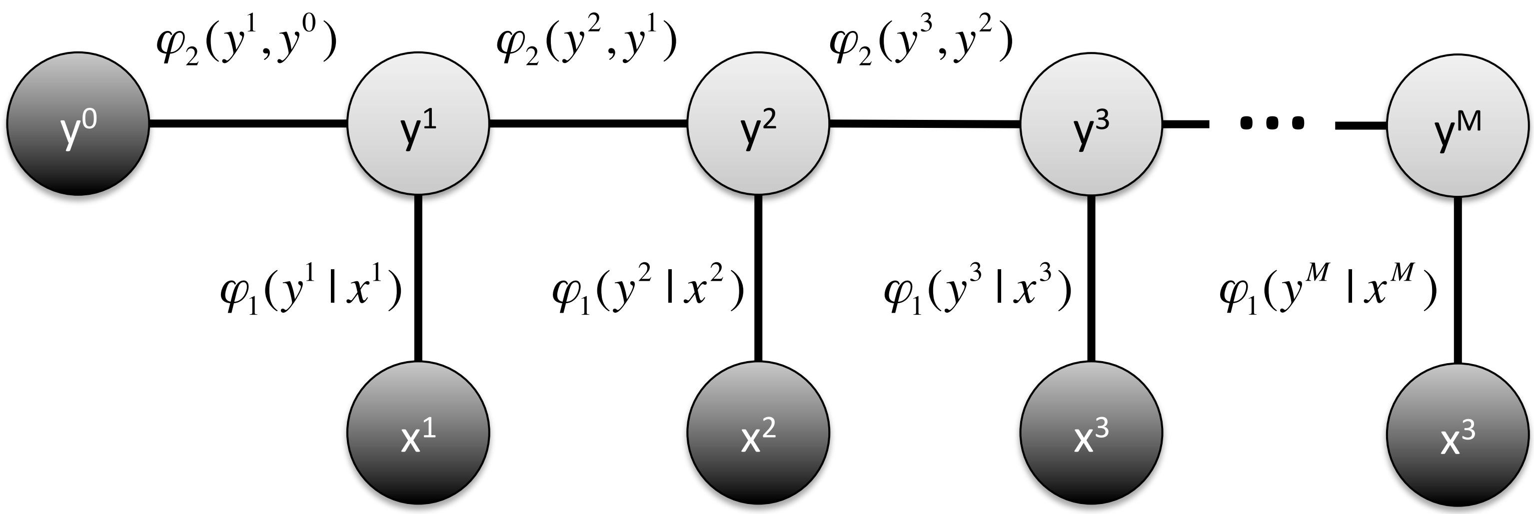
- Learning Objective: $\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$

Optimization Problem

Basic Unsupervised Learning

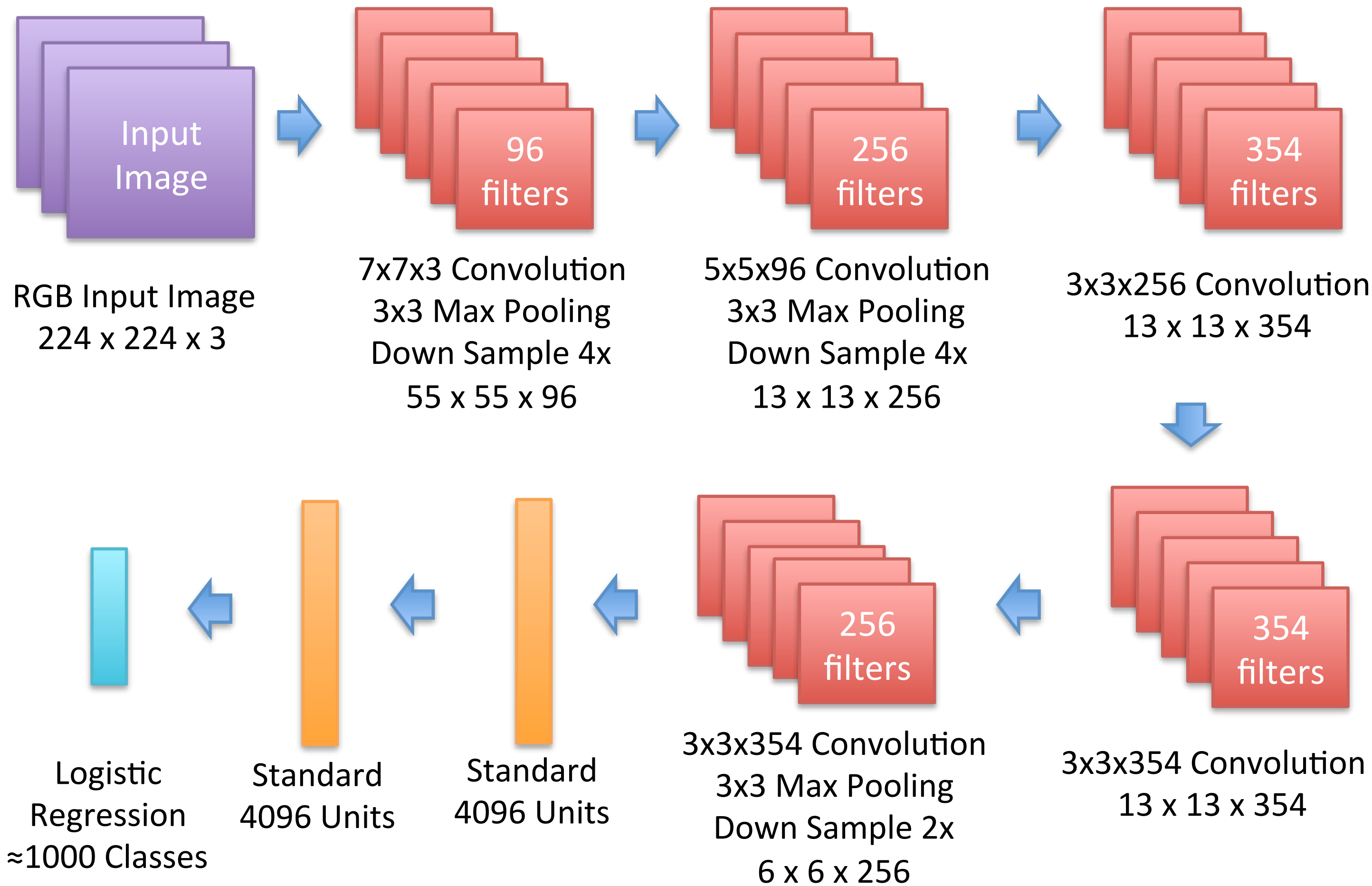


Sequence Prediction



$$\varphi^j(a,b|x) = \begin{bmatrix} \varphi_1^j(a|x) \\ \varphi_2(a,b) \end{bmatrix}$$

Intro to Deep Learning



Simple Optimization Algorithms

- Stochastic Gradient Descent

- EM algorithm (for HMMs)

Other Basic Concepts

- Cross Validation
- Overfitting
- Bias-Variance Tradeoff

Learning Theory

Generalization Bounds

- Formal characterization of overfitting
- Example result:

With Prob. ≥ 1-δ :

Test Error

Training Error

Make rigorous!

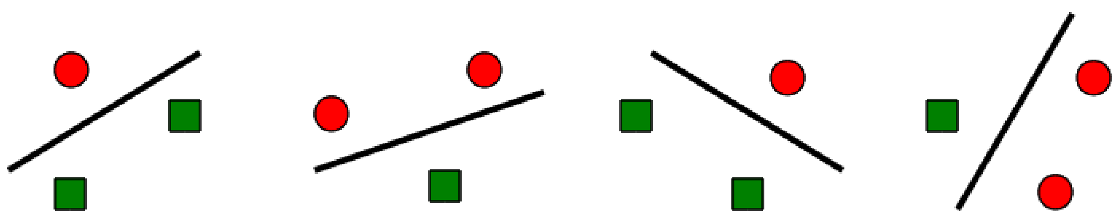
$$\underbrace{E_{out}(h)}_{\text{Test Error}} \leq \underbrace{E_{in}(h)}_{\text{Training Error}} + O\left(\frac{\log(1/\delta)}{\sqrt{N}}\right)$$

Trained Model

Training Size

Shattering

- **Definition:** A set of points is **shattered** by H if for all **possible binary labelings** of points, there exists some h that classifies perfectly.

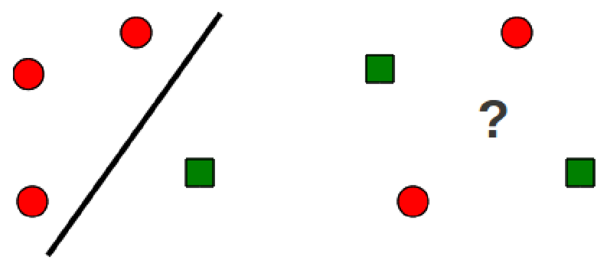


In 2D, any 3 points can always be shattered by linear models!

Slide Material Borrowed From Piyush Rai:
https://www.cs.utah.edu/~piyush/teaching/27-9-print.pdf

Shattering

- **Definition:** A set of points is **shattered** by H if for all **possible binary labelings** of points, there exists some h that classifies perfectly.



In 2D, linear models cannot shatter 4 points!

Slide Material Borrowed From Piyush Rai:
https://www.cs.utah.edu/~piyush/teaching/27-9-print.pdf

VC Dimension

- VC(H) = most # points that can be shattered
  - If H is linear models in 2D feature space:
    - VC(H) = 3

With Prob. ≥ 1-δ :

$$E_{out}(h) \leq E_{in}(h) + O\left(\frac{\log\left(\frac{2N}{VC(H)}+1\right)+\log\left(\frac{1}{\delta}\right)}{\sqrt{N}}\right)$$

Structured Prediction

Examples of Complex Output Spaces

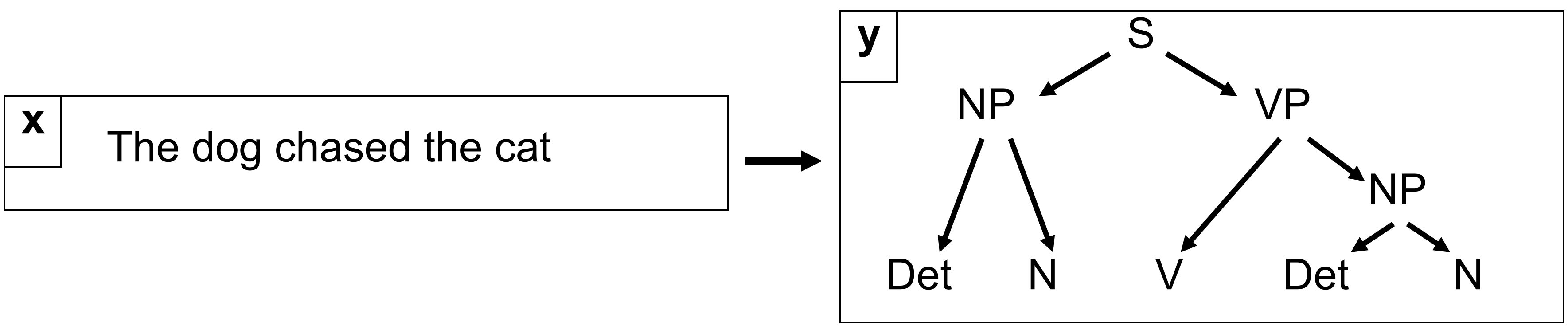
- Part-of-Speech Tagging
  - Given a sequence of words *x*, predict sequence of tags *y*.
  - Dependencies from tag-tag transitions in Markov model.

x The rain wet the cat → **y** Det → N → V → Det → N

→ Similarly for other sequence labeling problems, e.g., RNA Intron/ Exon Tagging.

Examples of Complex Output Spaces

- Natural Language Parsing
  - Given a sequence of words *x*, predict the parse tree *y*.
  - Dependencies from structural constraints, since *y* has to be a tree.



Examples of Complex Output Spaces

- Information Retrieval
  - Given a query x, predict a ranking *y*.
  - Dependencies between results (e.g. avoid redundant hits)
  - Loss function over rankings (e.g. Average Precision)

x

SVM

→

y

1. **Kernel-Machines**
2. **SVM-Light**
3. **Learning with Kernels**
4. **SV Meppen Fan Club**
5. **Service Master & Co.**
6. **School of Volunteer Management**
7. **SV Mattersburg Online**

…

General Formula (Linear Models)

- Assume scoring function F

$$h(\mathbf{x};w) = \operatorname*{argmax}_{\mathbf{y}\in Y(\mathbf{x})} F(\mathbf{x},\mathbf{y};w)$$

- Assume F is linear:

$$F(\mathbf{x},\mathbf{y};w) = w^T \Psi(\mathbf{x},\mathbf{y})$$

Example 1

$$h(\mathbf{x}; w) = \underset{\mathbf{y} \in Y(\mathbf{x})}{\operatorname{argmax}} F(\mathbf{x}, \mathbf{y}; w) \qquad F(\mathbf{x}, \mathbf{y}; w) = w^T \Psi(\mathbf{x}, \mathbf{y})$$

Binary Classification:

$$Y(\mathbf{x}) = \{-1, +1\}$$

$$\Psi(\mathbf{x}, y) = y\mathbf{x}$$

$$F(\mathbf{x}, y; w) = y(w^T \mathbf{x})$$

$$h(\mathbf{x}; w) = \underset{y \in \{-1, +1\}}{\operatorname{argmax}} y\left(w^T \mathbf{x}\right)$$

Examples

$$h(\mathbf{x};w) = \operatorname*{argmax}_{\mathbf{y}\in Y(\mathbf{x})} F(\mathbf{x},\mathbf{y};w)$$

$$F(\mathbf{x},\mathbf{y};w) = w^T \Psi(\mathbf{x},\mathbf{y})$$

1st Order Sequences:

$Y(\mathbf{x}) =$ all possible output sequences

$$\Psi(\mathbf{x},\mathbf{y}) = \sum_j \phi(y^j, y^{j-1} \mid \mathbf{x})$$

$$F(\mathbf{x},\mathbf{y};w) = w^T \sum_j \phi(y^j, y^{j-1} \mid \mathbf{x})$$

Solve using Viterbi!

Examples

$$h(\mathbf{x};w) = \underset{\mathbf{y}\in Y(\mathbf{x})}{\operatorname{argmax}} F(\mathbf{x},\mathbf{y};w)$$

$$F(\mathbf{x},\mathbf{y};w) = w^T \Psi(\mathbf{x},\mathbf{y})$$

Integer Linear Program:

$Y(\mathbf{x}) =$ Feasible settings of **y**

Each $y^j \in \{0,1\}$

$$\Psi(\mathbf{x},\mathbf{y}) = \sum_j y^j \phi^j(\mathbf{x})$$

$$F(\mathbf{x},\mathbf{y};w) = \mathbf{y}^T\mathbf{c} \qquad \mathbf{c} = \begin{bmatrix} w^T\phi^1(\mathbf{x}) \\ w^T\phi^2(\mathbf{x}) \\ \vdots \end{bmatrix}$$

$$h(\mathbf{x};w) = \underset{\mathbf{y}\in Y(\mathbf{x})}{\operatorname{argmax}} \mathbf{y}^T\mathbf{c}$$

Structured Prediction Learning Problem

- **Efficient Inference/Prediction**

$$\boxed{h(\mathbf{x}; w) = \operatorname*{argmax}_{\mathbf{y}} w^T \Psi(\mathbf{y}, \mathbf{x})}$$

  - Viterbi in sequence labeling
  - CKY Parser for parse trees
  - Sorting for ranking

- **Efficient Learning/Training**
  - Learn parameters *w* from training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1..N}$
  - Structural SVM: Hinge Loss Minimization
  - Conditional Random Fields: Log Loss Minimization
  - Structured Perceptron, etc…

Perceptron Learning Algorithm

- $w^1 = 0$, $b^1 = 0$
- For t = 1 ….
  - Receive example (x,y)
  - If $h(x|w^t) = y$
    - $[w^{t+1}, b^{t+1}] = [w^t, b^t]$
  - Else
    - $w^{t+1} = w^t + yx$
    - $b^{t+1} = b^t + y$

$$h(x\,|\,w) = sign(w^T x - b)$$

Training Set:

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

$$y \in \{+1, -1\}$$

Go through training set in arbitrary order (e.g., randomly)

Structured Perceptron

$$h(x \mid w) = \underset{y'}{\operatorname{argmax}}\, w^T \Psi(x, y')$$

- $w^1 = 0$
- For t = 1 ….
  - Receive example (x,y)
  - If $h(x|w^t) = y$
    - $w^{t+1} = w^t$
  - Else
    - $w^{t+1} = w^t + \Psi(x,y)$

Training Set:

$$S = \{(x_i, y_i)\}_{i=1}^{N}$$

Go through training set in arbitrary order (e.g., randomly)

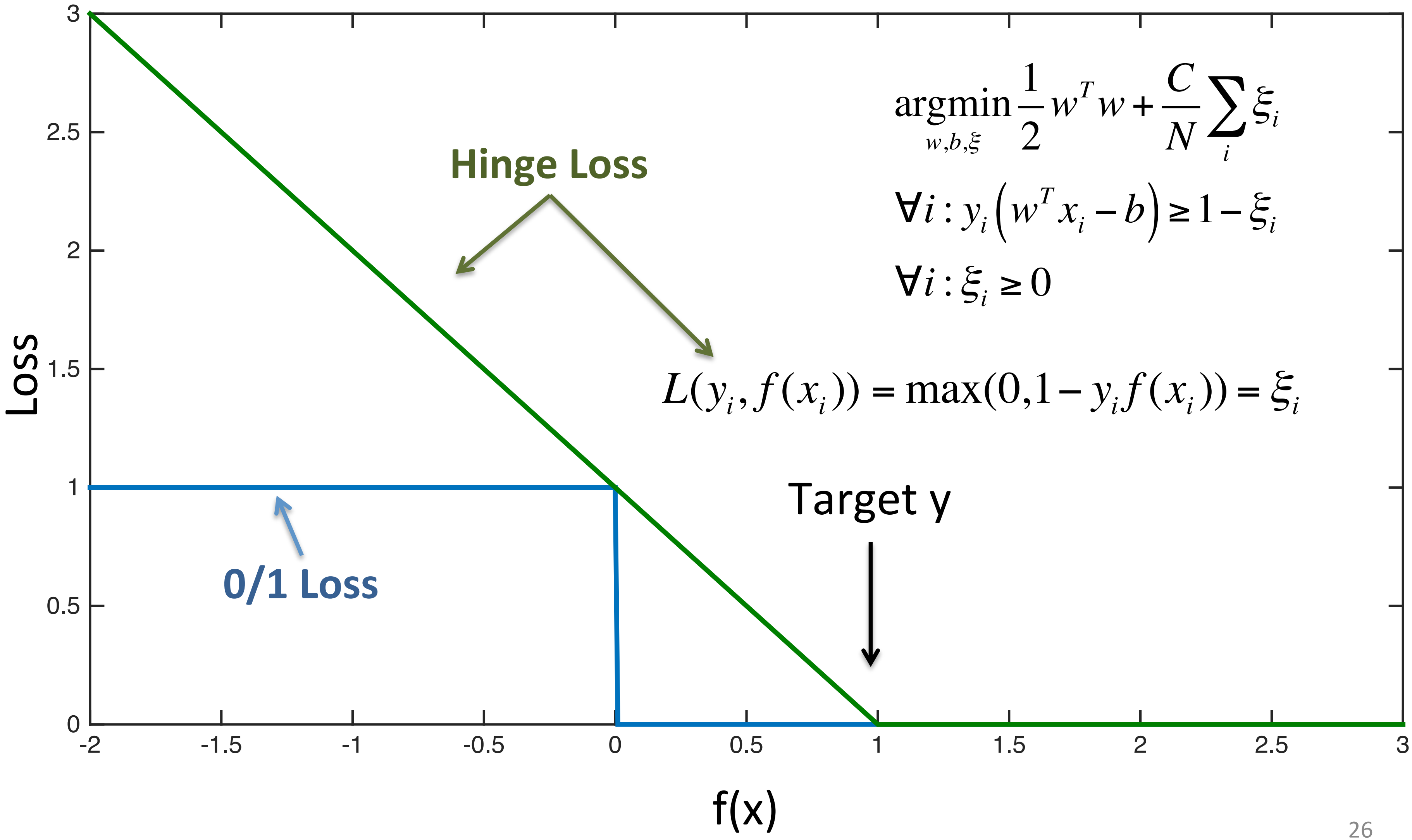
Conventional SVMs

- Input: x (high dimensional point)
- Target: y (either +1 or -1)
- Prediction: sign($w^T x$)
- Training: $$\arg\min_{w,\xi} \frac{1}{2} w^2 + \frac{C}{N} \sum_{i=1}^{N} \xi_i$$

  subject to: $$\forall i: \quad y_i \cdot (w^T x_i) \geq 1 - \xi_i$$

- **The sum of slacks $\sum_i \xi_i$ upper bounds the 0/1 loss!**

Conventional SVMs



Structural SVM

- Let **x** denote a structured input (sentence)
- Let **y** denote a structured output (POS tags)

- Standard objective function: $\frac{1}{2}w^2 + \frac{C}{N}\sum_i \xi_i$

- Constraints are defined for each incorrect labeling **y'** over each **x**.

$$\forall i, \forall \mathbf{y}' \neq \mathbf{y}^{(i)}: \quad \underbrace{w^T \Psi(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})}_{\textbf{Score}(\mathbf{y}^{(\mathbf{i})})} \geq \underbrace{w^T \Psi(\mathbf{y}', \mathbf{x}^{(i)})}_{\textbf{Score}(\mathbf{y}')} + \underbrace{\Delta_i(\mathbf{y}')}_{\textbf{Loss}(\mathbf{y}')} - \underbrace{\xi_i}_{\textbf{Slack}}$$

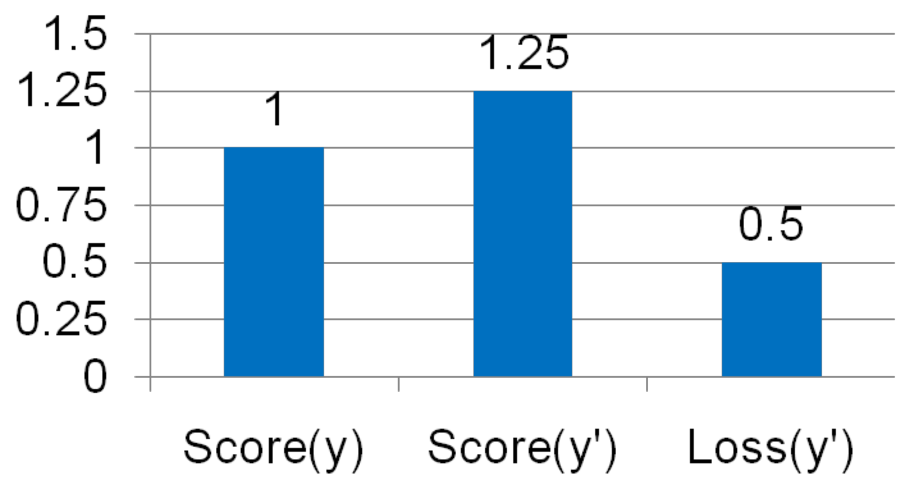
[Tsochantaridis et al., 2005]

Interpreting Constraints

$$\frac{1}{2}w^2 + \frac{C}{N}\sum_i \xi_i$$

$$\forall i, \forall \mathbf{y}' \neq \mathbf{y}^{(i)} : \quad \underbrace{w^T \Psi(\mathbf{y}^{(i)}, \mathbf{x}^{(i)})}_{\textbf{Score}(\mathbf{y}^{(i)})} \geq \underbrace{w^T \Psi(\mathbf{y}', \mathbf{x}^{(i)})}_{\textbf{Score}(\mathbf{y}')} + \underbrace{\Delta_i(\mathbf{y}')}_{\textbf{Loss}(\mathbf{y}')} - \underbrace{\xi_i}_{\textbf{Slack}}$$

Suppose for incorrect **y'** :



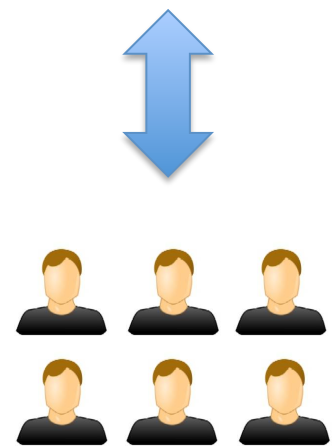
Then: $\boxed{\xi_i \geq 0.75 \geq \Delta(\mathbf{y}')}$

[Tsochantaridis et al., 2005]

Crowdsourcing

Acquiring Labels from Annotators

amazon mechanical turk beta



How Reliable are Annotators?

- If we knew what the labels were
  - Can judge workers on label quality
- If we knew who the good workers were
  - Can create labels from their annotations
- **Chicken and egg problem!**

Worker Reliability as Latent Variable

- Let z_m denote the reliability of worker m

Estimated label

$$y_i = \frac{1}{\sum_m z_m} \sum_m y_{im} z_m$$

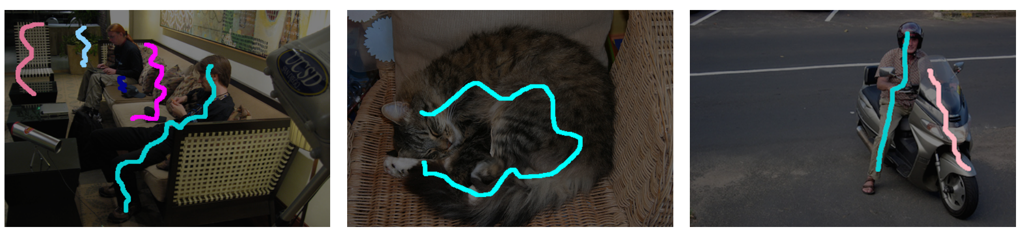
$$z_m = \frac{1}{N} \sum_i L(y_i, y_{im})$$

Differing Ambiguities Across Tasks

- Often collecting annotations for many tasks
- Some tasks are harder than others
- How many labels to collect for each task?

Structured Annotations

Full supervision | Image-level supervision | Point-level supervision



http://arxiv.org/pdf/1506.02106v4.pdf

Active Learning

Crowdsourcing

Unlabeled

Repeat

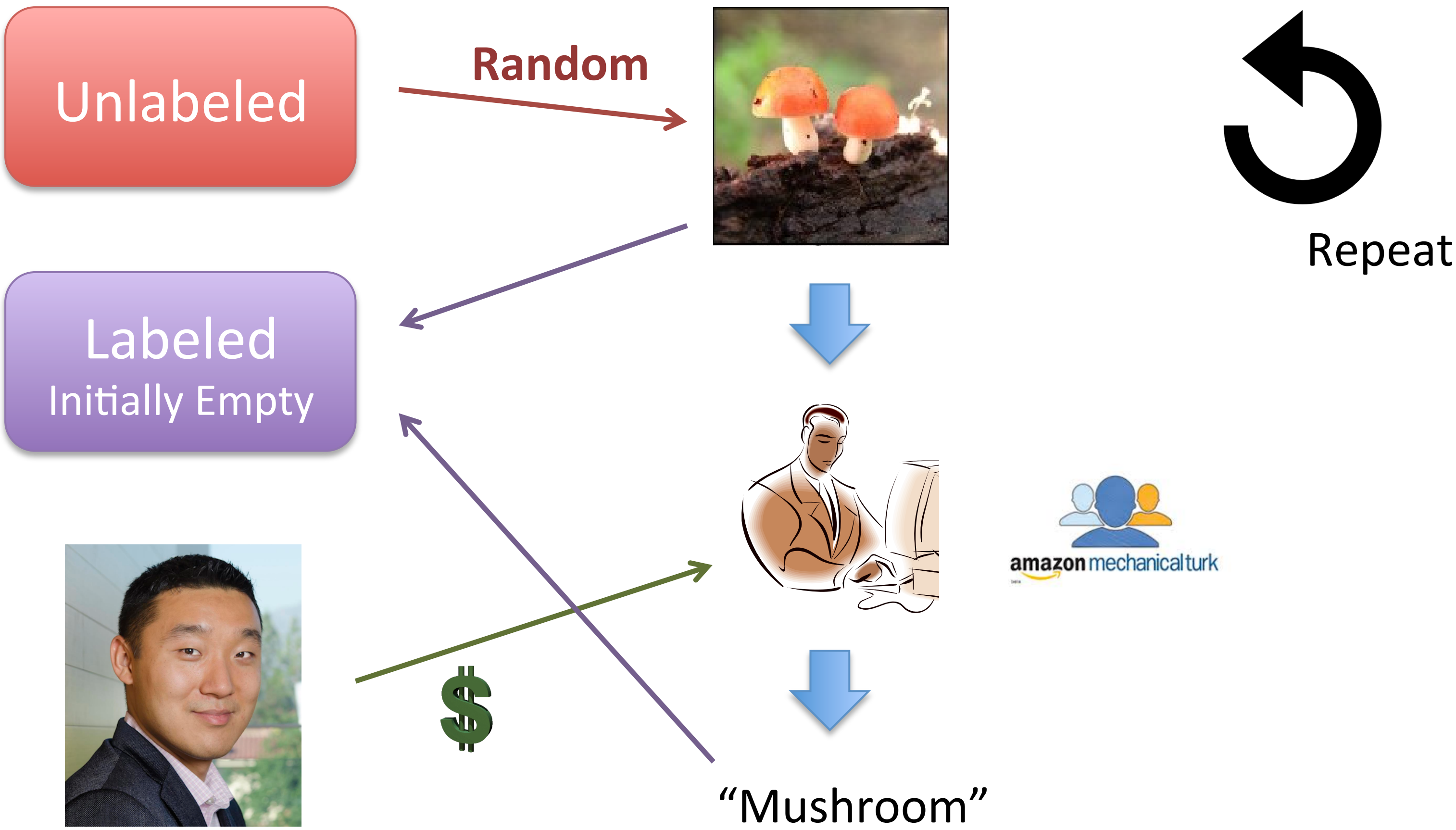
Labeled
Initially Empty

amazon mechanicalturk

$

“Mushroom”

Passive Learning



Active Learning

Unlabeled

Choose

Repeat

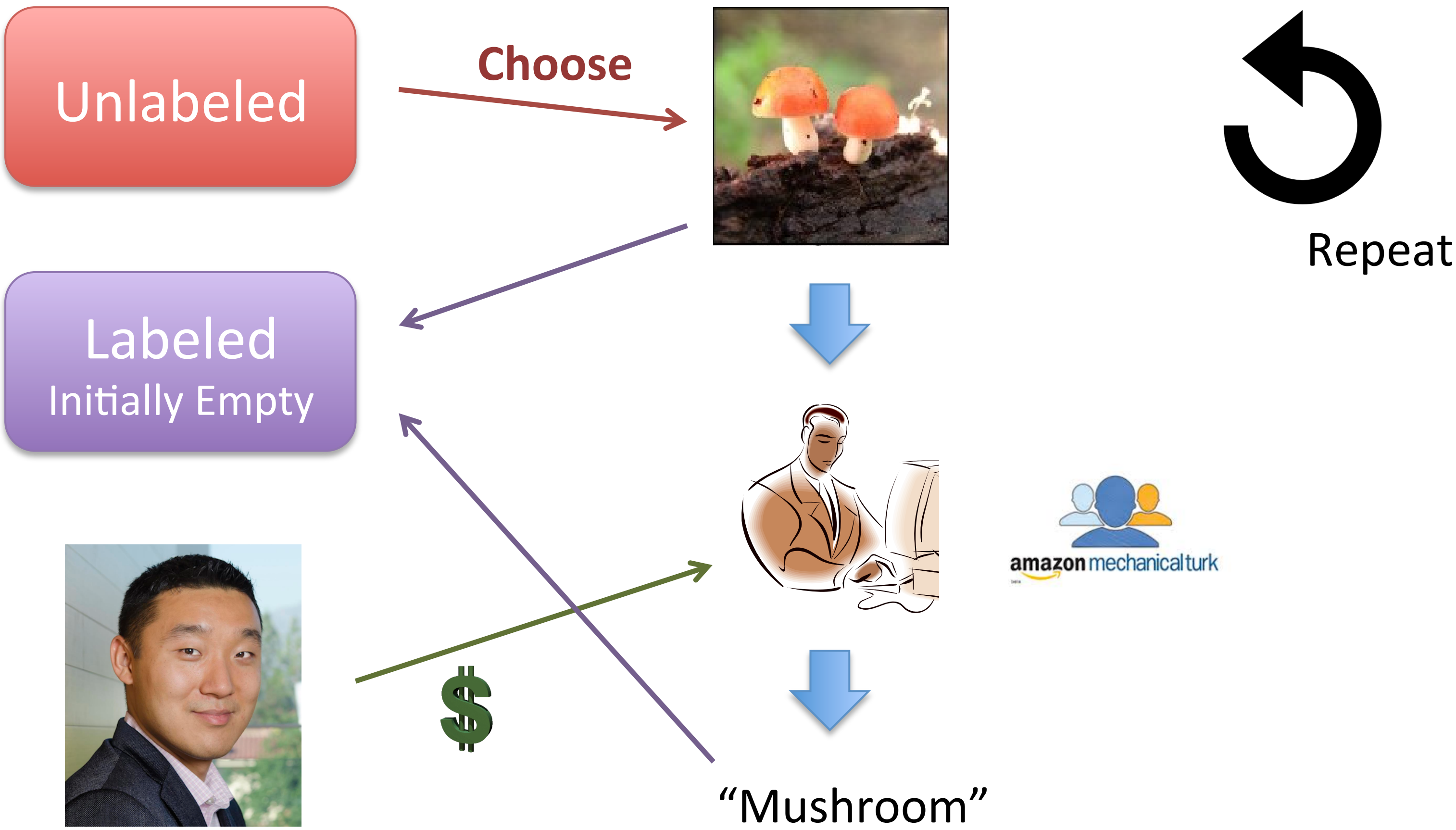
Labeled
Initially Empty

amazon mechanicalturk

$

"Mushroom"

Goal: Maximize Accuracy with Minimal Cost



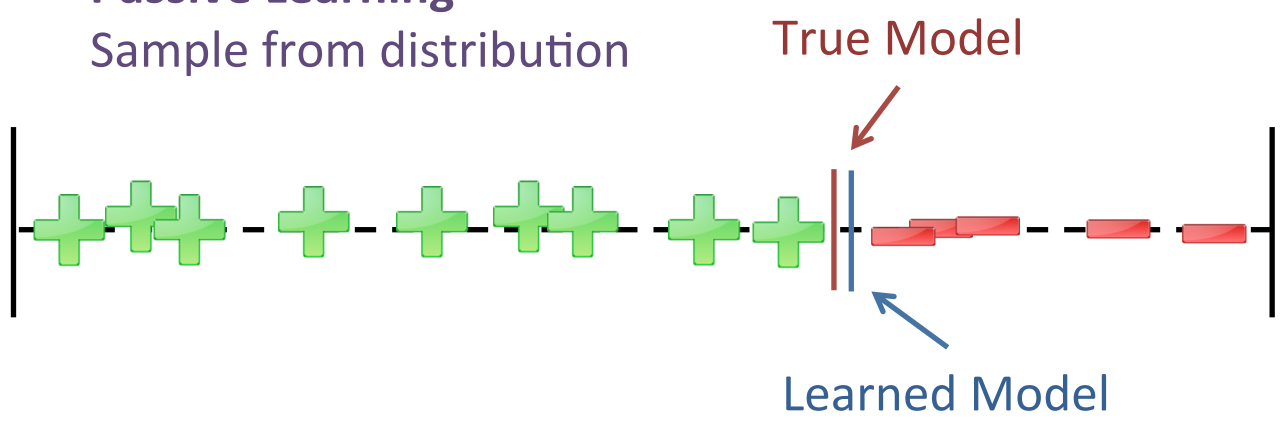
Comparison with Passive Learning

- Conventional Supervised Learning is considered "Passive" Learning
- Unlabeled training set sampled according to test distribution
- So we label it at random
  - **Very Expensive!**

Simple Example

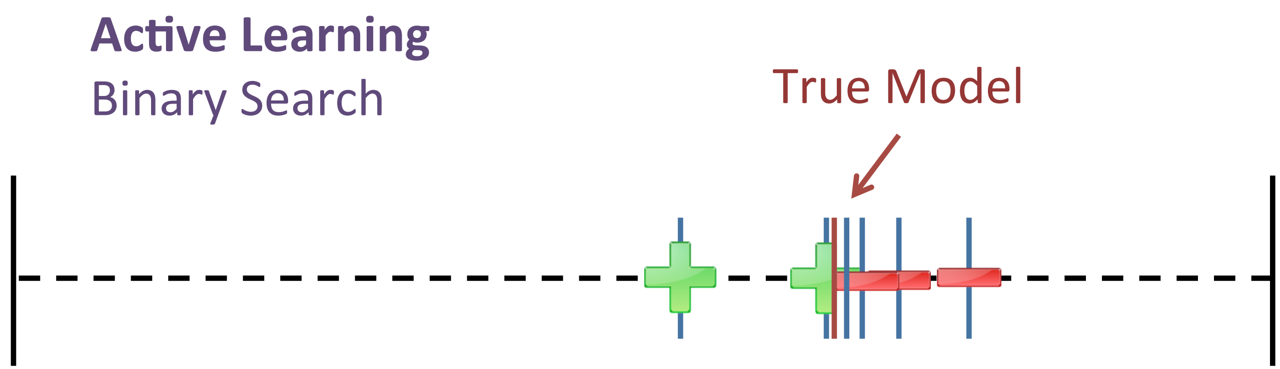
- 1 feature
- Learn threshold function

Passive Learning
Sample from distribution



Simple Example

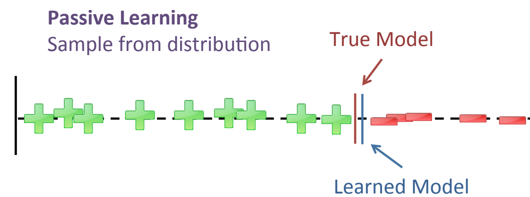
- 1 feature
- Learn threshold function



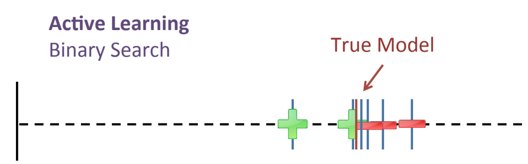
Comparison with Passive Learning

- # samples to be within ε of true model

- Passive Learning: $O\left(\frac{1}{\varepsilon}\right)$



- Active Learning: $O\left(\log\frac{1}{\varepsilon}\right)$

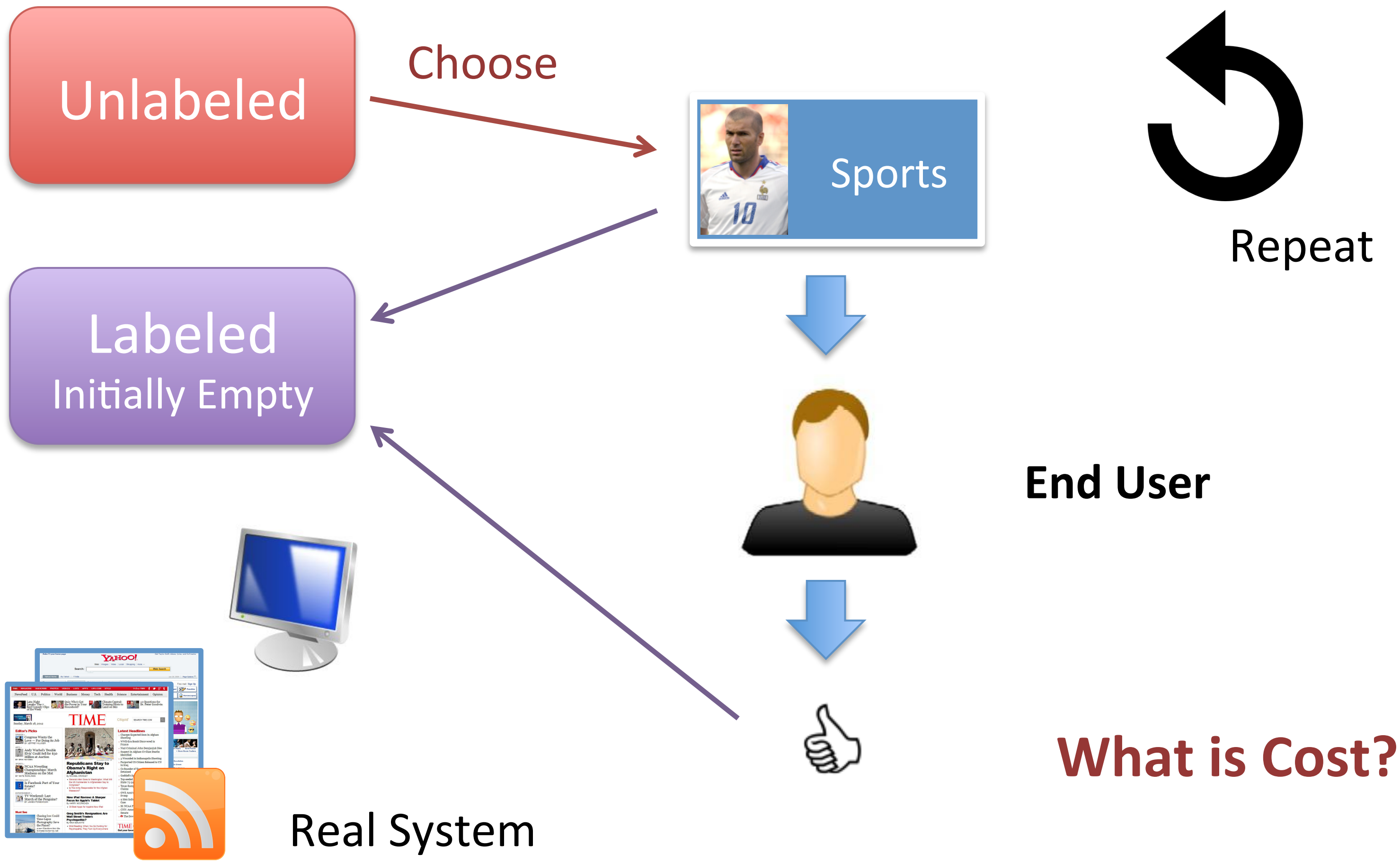


Multi-Armed Bandits

Problems with Crowdsourcing

- Assumes you can label by proxy
  - E.g., have someone else label objects in images
- But sometimes you can't!
  - Personalized recommender systems
    - Need to ask the user whether content is interesting
  - Personalized medicine
    - Need to try treatment on patient
  - **Requires actual target domain**

Personalized Labels



Formal Definition

- K actions/classes
- Each action has an average reward: μ_k
  - Unknown to us
  - Assume WLOG that u_1 is largest

Basic Setting
K classes
No features

- For t = 1…T
  - Algorithm chooses action a(t)
  - Receives random reward y(t)
    - Expectation $\mu_{a(t)}$

Algorithm Simultaneously
Predicts & Receives Labels

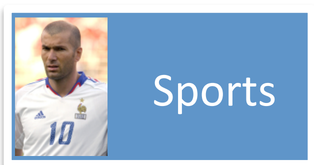
- **Goal:** minimize $Tu_1 - (\mu_{a(1)} + \mu_{a(2)} + \ldots + \mu_{a(T)})$

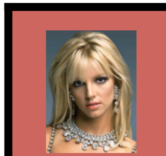
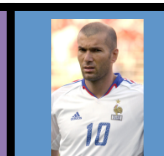
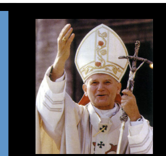
If we had perfect information to start

Expected Reward of Algorithm

Interactive Personalization

(5 Classes, No features)

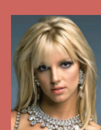


					
Average Likes	--	--	--	--	--
# Shown	0	0	0	1	0

 : 0

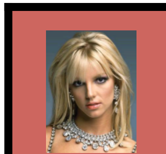
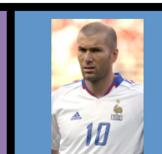
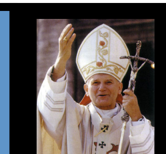
Interactive Personalization

(5 Classes, No features)

					
Average Likes	--	--	--	0	--
# Shown	0	0	0	1	0

 : 0

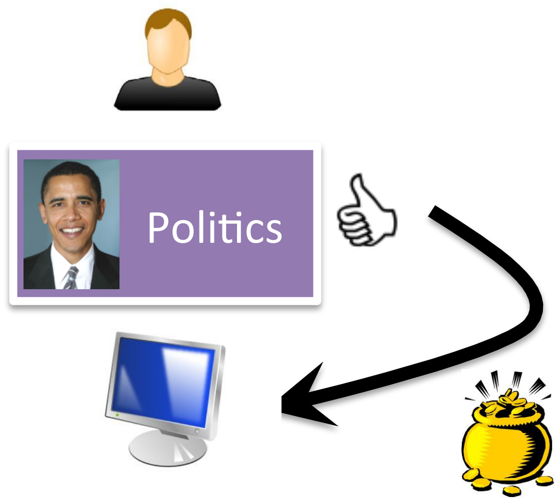
Interactive Personalization
(5 Classes, No features)

					
Average Likes	--	--	--	0	--
# Shown	0	0	1	1	0

 : 0

Interactive Personalization

(5 Classes, No features)



Average Likes	--	--	1	0	--
# Shown	0	0	1	1	0

 : 1

Interactive Personalization

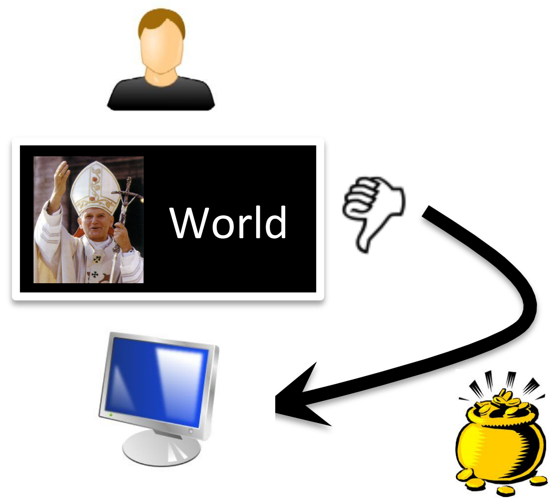
(5 Classes, No features)

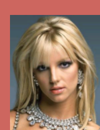
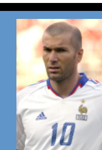
					
Average Likes	--	--	1	0	--
# Shown	0	0	1	1	1

 : 1

Interactive Personalization

(5 Classes, No features)



					
Average Likes	--	--	1	0	0
# Shown	0	0	1	1	1

 : 1

Interactive Personalization

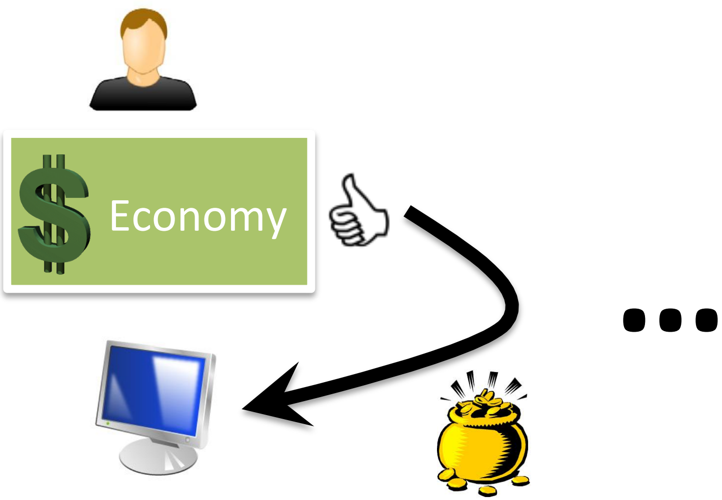
(5 Classes, No features)

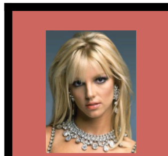
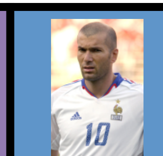
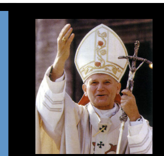
Average Likes	--	--	1	0	0
# Shown	0	1	1	1	1

 : 1

Interactive Personalization

(5 Classes, No features)



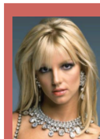
					
Average Likes	--	1	1	0	0
# Shown	0	1	1	1	1

What should Algorithm Recommend?

Exploit:

 Economy

Explore:

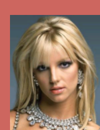
 Celebrity

Best:

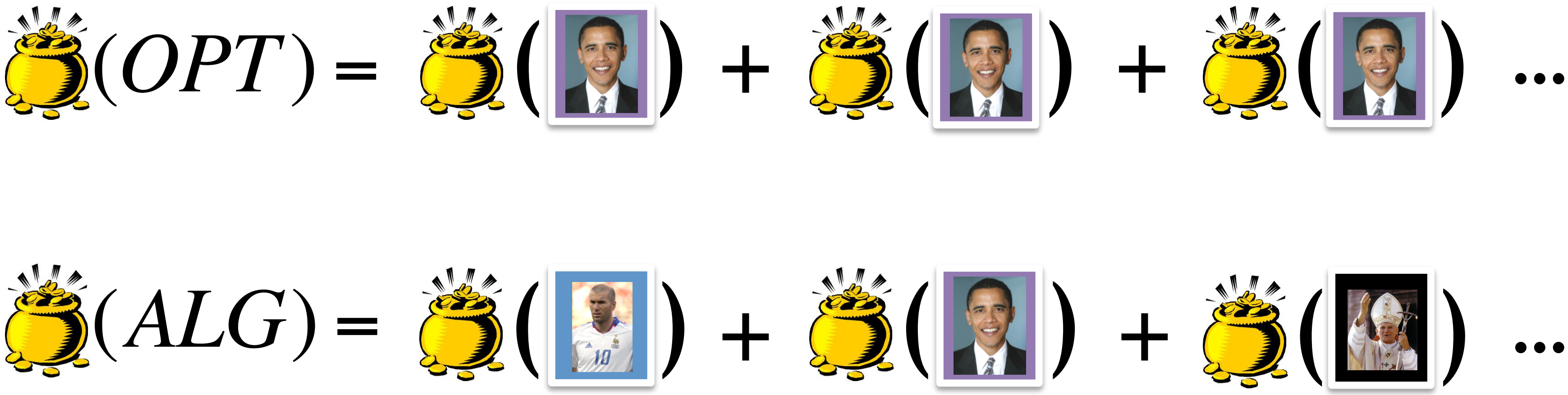
 Politics

How to Optimally Balance Explore/Exploit Tradeoff?

Characterized by the Multi-Armed Bandit Problem

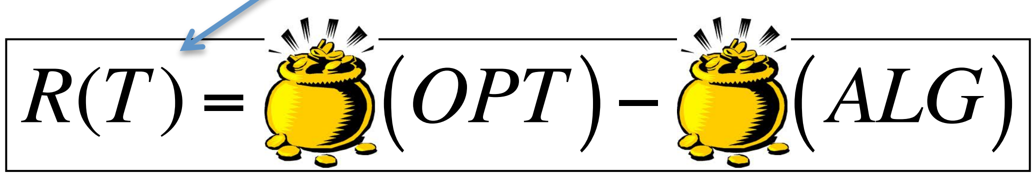
					
Average Likes	--	0.44	0.4	0.33	0.2
# Shown	0	25	10	15	20

 : 24



Time Horizon

Regret:



- Opportunity cost of not knowing preferences
- **“no-regret”** if R(T)/T → 0
  - Efficiency measured by convergence rate

Recap: The Multi-Armed Bandit Problem

- K actions/classes
- Each action has an average reward: μ_k
  - All unknown to us
  - Assume WLOG that u_1 is largest

Basic Setting
K classes
No features

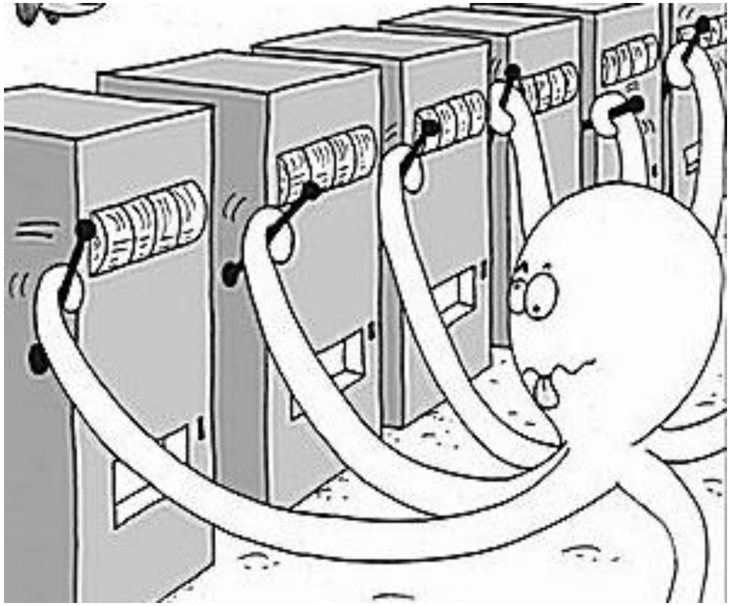
- For t = 1…T
  - Algorithm chooses action a(t)
  - Receives random reward y(t)
    - Expectation $\mu_{a(t)}$

Algorithm Simultaneously
Predicts & Receives Labels

- Goal: minimize $\underbrace{Tu_1 - (\mu_{a(1)} + \mu_{a(2)} + \ldots + \mu_{a(T)})}_{\text{Regret}}$

The Motivating Problem

- Slot Machine = One-Armed Bandit



Each Arm Has
Different Payoff

- **Goal:** Minimize regret From pulling suboptimal arms

http://en.wikipedia.org/wiki/Multi-armed_bandit

Implications of Regret

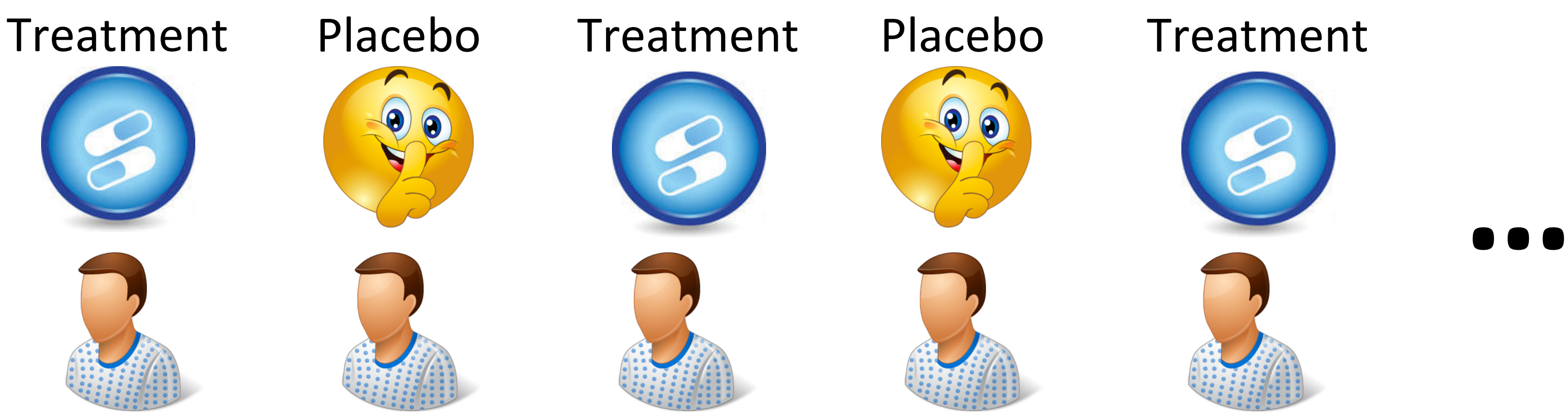
Regret: $R(T) =$  $(OPT) -$  (ALG)

- If R(T) grows linearly w.r.t. T:
  - Then R(T)/T ➔ constant > 0
  - I.e., we converge to predicting something suboptimal

- If R(T) is sub-linear w.r.t. T:
  - Then R(T)/T ➔ 0
  - I.e., we converge to predicting the optimal action

Experimental Design

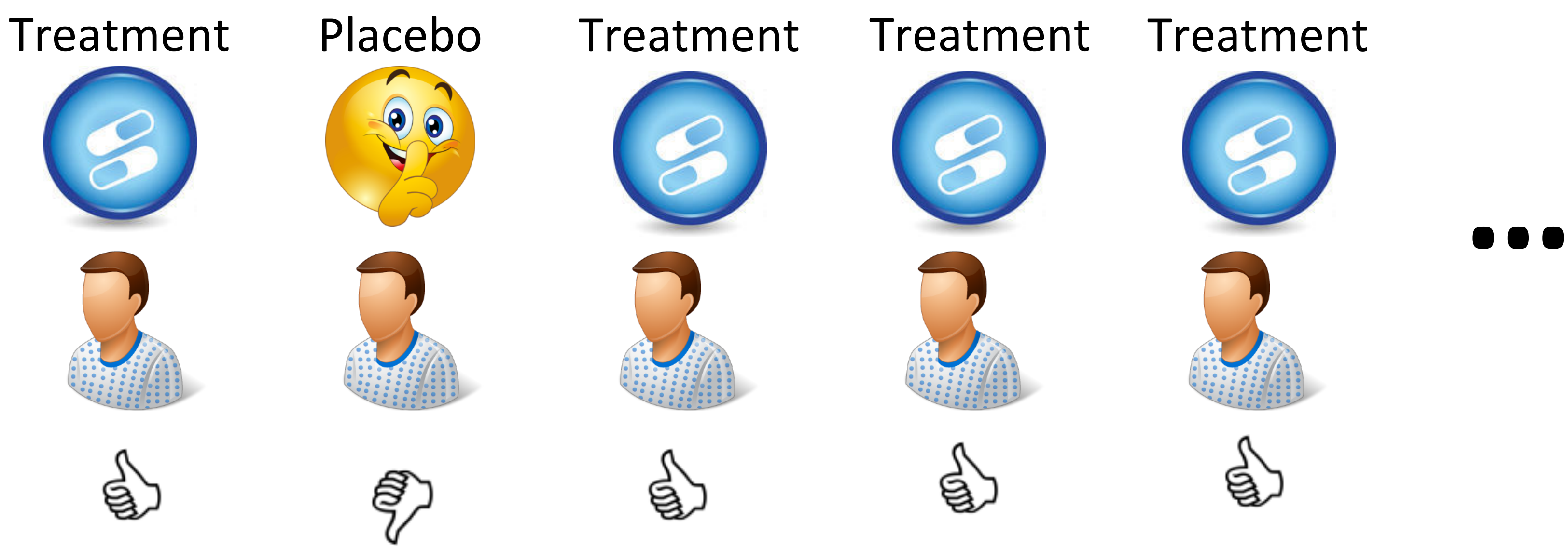
- How to split trials to collect information
- **Static Experimental Design**
  - Standard practice
  - (pre-planned)



http://en.wikipedia.org/wiki/Design_of_experiments

Sequential Experimental Design

- Adapt experiments based on outcomes



Sequential Experimental Design Matters

Monica Almeida/The New York Times, left

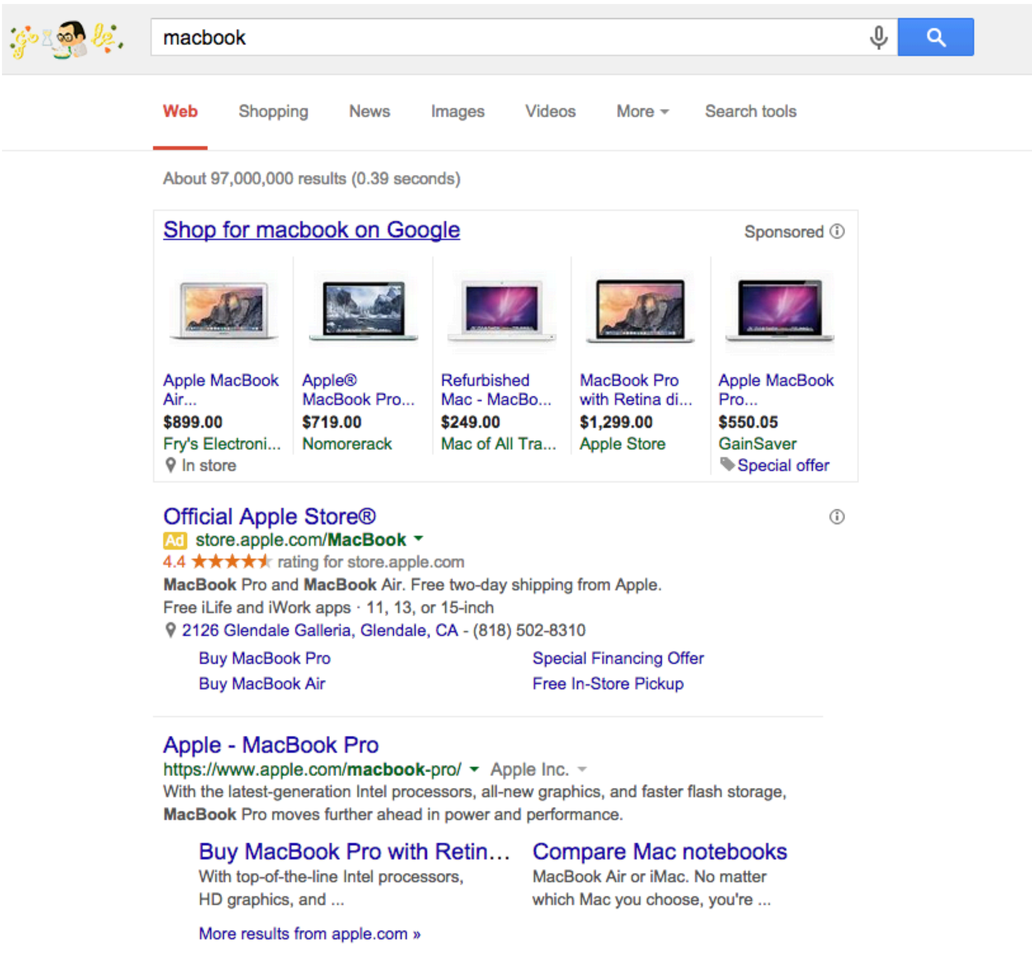
Two Cousins, Two Paths Thomas McLaughlin, left, was given a promising experimental drug to treat his lethal skin cancer in a medical trial; Brandon Ryan had to go without it.

http://www.nytimes.com/2010/09/19/health/research/19trial.html

Sequential Experimental Design

- MAB models sequential experimental design!
  basic
- Each treatment has hidden expected value
  - Need to run trials to gather information
  - "Exploration"
- In hindsight, should always have used treatment with highest expected value
- **Regret = opportunity cost of exploration**

Online Advertising



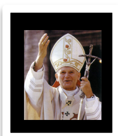
Largest Use-Case of Multi-Armed Bandit Problems

The UCB1 Algorithm

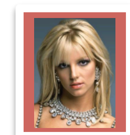
http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

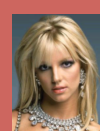
Confidence Intervals

- Maintain Confidence Interval for Each Action
  - Often derived using Chernoff-Hoeffding bounds (**)

 = [0.1, 0.3]

 = [0.25, 0.55]

 Undefined

					
Average Likes	--	0.44	0.4	0.33	0.2
# Shown	0	25	10	15	20

** http://www.cs.utah.edu/~jeffp/papers/Chern-Hoeff.pdf
http://en.wikipedia.org/wiki/Hoeffding%27s_inequality

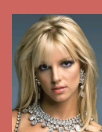
UCB1 Confidence Interval

Expected Reward Estimated from data

$$\bar{\mu}_k \pm \sqrt{\frac{2\ln t}{t_k}}$$

Total Iterations so far (70 in example below)

#times action k was chosen

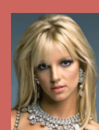
					
Average Likes	--	0.44	0.4	0.33	0.2
# Shown	0	25	10	15	20

http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

The UCB1 Algorithm

- At each iteration
  - Play arm with highest **Upper Confidence Bound**:

$$\underset{k}{\operatorname{argmax}}\ \bar{\mu}_k + \sqrt{(2\ln t)/t_k}$$

					
Average Likes	--	0.44	0.4	0.33	0.2
# Shown	0	25	10	15	20

http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

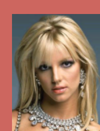
Balancing Explore/Exploit

"Optimism in the Face of Uncertainty"

$$\underset{k}{\operatorname{argmax}}\ \bar{\mu}_k + \sqrt{(2\ln t)/t_k}$$

Exploitation Term

Exploration Term

					
Average Likes	--	0.44	0.4	0.33	0.2
# Shown	0	25	10	15	20

http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

Analysis (Intuition)

$$a(t+1) = \underset{k}{\operatorname{argmax}}\, \bar{\mu}_k + \sqrt{(2\ln t)/t_k}$$

With high probability (**):

$$\bar{\mu}_{a(t+1)} + \sqrt{(2\ln t)/t_{a(t+1)}} \geq \overbrace{\bar{\mu}_1 + \sqrt{(2\ln t)/t_1}}^{\text{Upper Confidence Bound of Best Arm}} \geq \mu_1$$

Value of Best Arm

$$\mu_{a(t+1)} \geq \bar{\mu}_{a(t+1)} - \sqrt{(2\ln t)/t_{a(t+1)}}$$

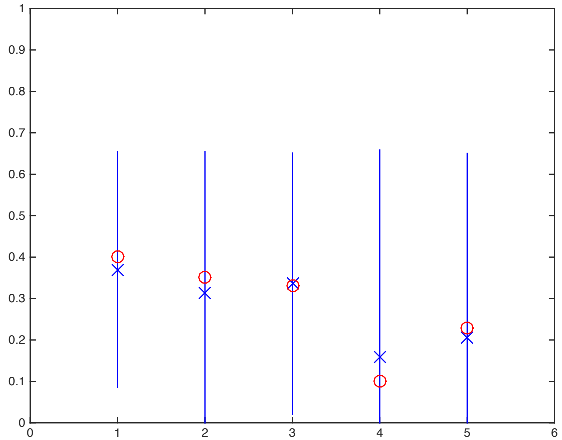
The true value is greater than the lower confidence bound.

$$\mu_1 - \mu_{a(t+1)} \leq 2\sqrt{(2\ln t)/t_{a(t+1)}}$$

Bound on regret at time t+1

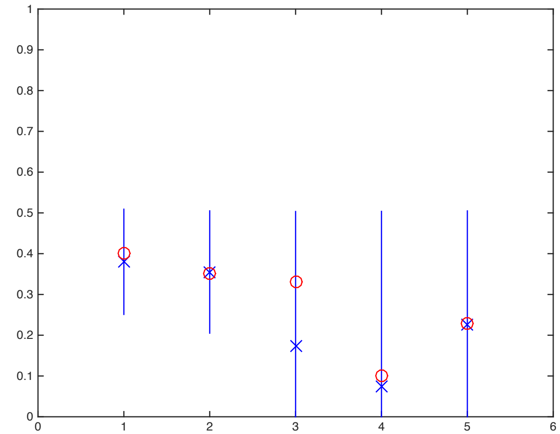
** Proof of Theorem 1 in http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

500 Iterations



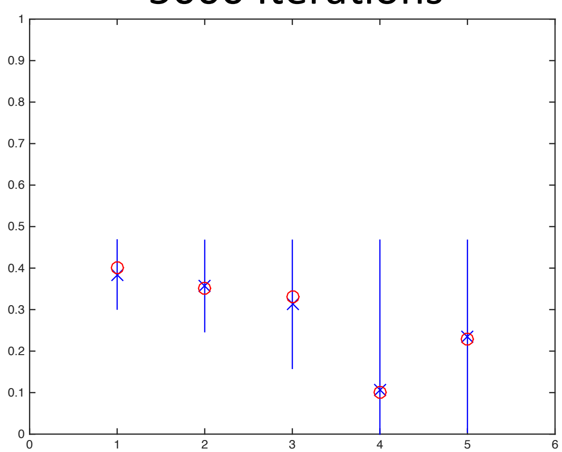
158 145 89 34 74

2000 Iterations



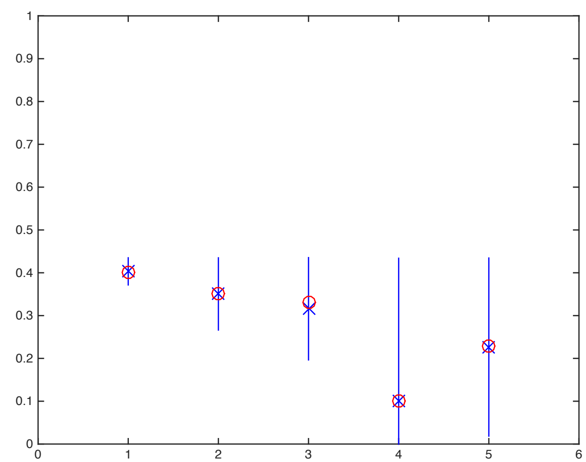
913 676 139 82 195

5000 Iterations



2442 1401 713 131 318

25000 Iterations



20094 2844 1418 181 468

How Often Sub-Optimal Arms Get Played

- An arm never gets selected if:

$$\mu_k + \sqrt{(2 \ln t) / t_k} \leq \mu_1$$

Bound grows slowly with time

Shrinks quickly with #trials

- The number of times selected: $O\left(\frac{\ln t}{(\mu_1 - \mu_k)^2}\right)$
  - Prove using Hoeffding's Inequality

Theorem 1 in http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

Regret Guarantee

- With high probability:
  - UCB1 accumulates regret at most:

#Actions

$$R(T) = O\left(\frac{K}{\varepsilon}\ln T\right)$$

Time Horizon

Gap between best & 2nd best

$\varepsilon = \mu_1 - \mu_2$

Theorem 1 in http://homes.di.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf

Extensions

- Contextual Bandits
  - Features of environment
- Dependent-Arms Bandits
  - Features of actions/classes
- Dueling Bandits
  - Learn from pairwise feedback

Recap: MAB & UCB1

- Interactive setting
  - Receives reward/label while making prediction
- Must balance explore/exploit
- Sub-linear regret is good
  - Average regret converges to 0

Reinforcement Learning

Actions Impact State

- In MAB:
  - Actions do not impact state
  - Constant reward function

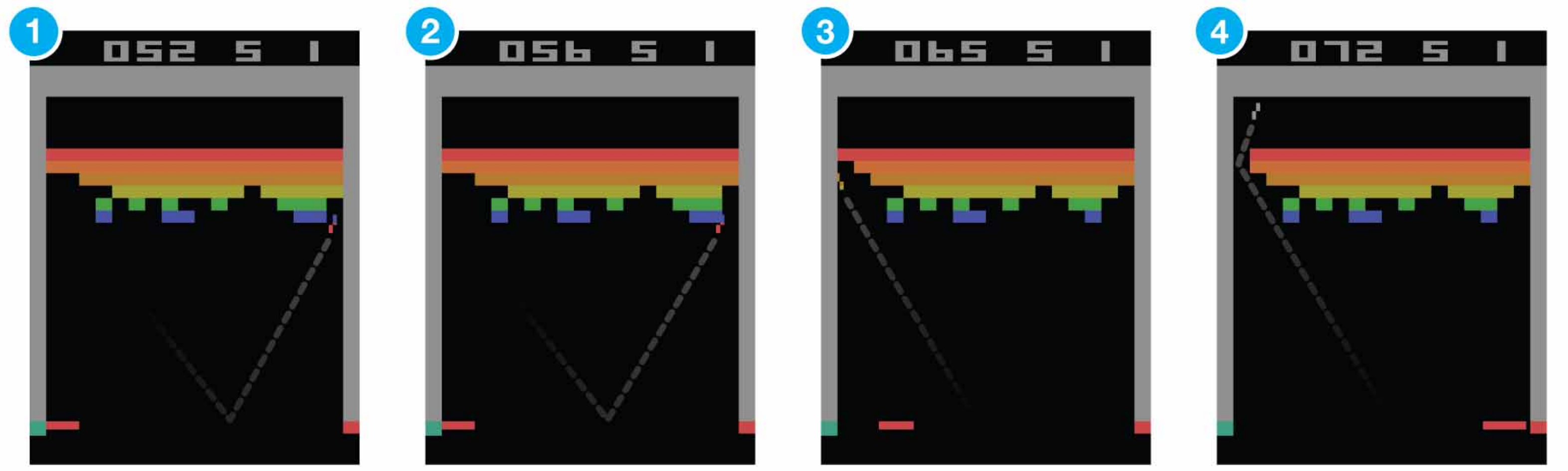
- Reinforcement Learning
  - Actions effect state you're in
  - Reward function depends on state

Video Demo

(Deep Reinforcement Learning for Atari)

https://www.youtube.com/watch?v=iqXKQf2BOSE

What is State?



Reward of each action varies depending on state!

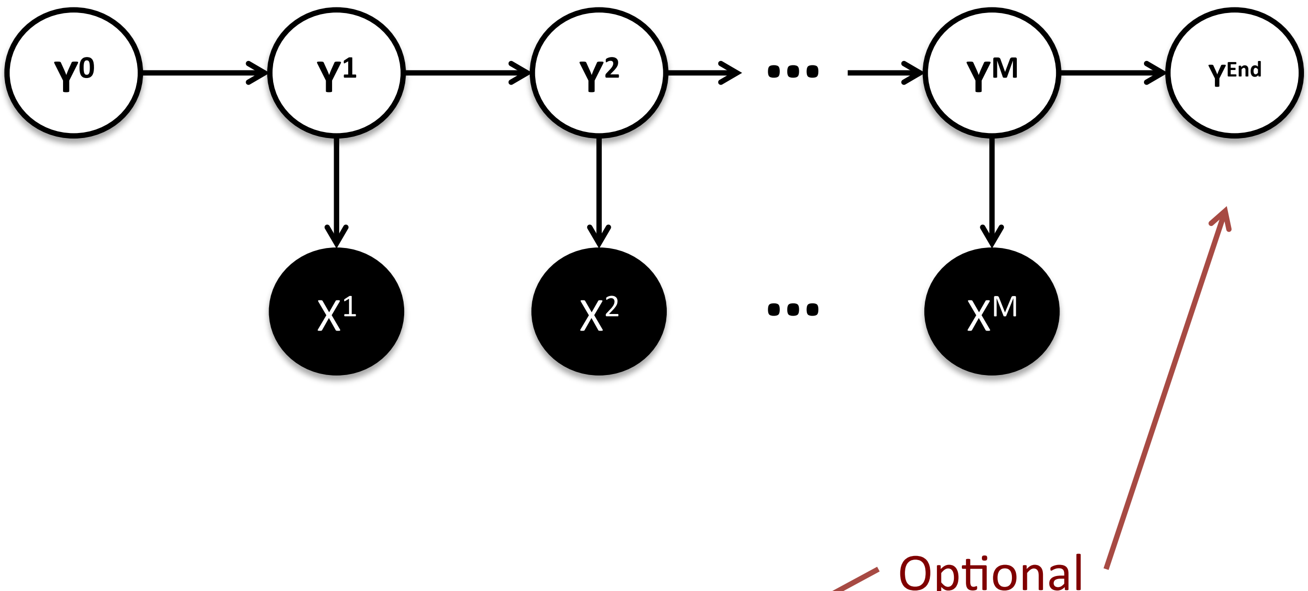
Action at current state impacts future states!

Much harder to do exploration!

http://www.nature.com/nature/journal/v518/n7540/pdf/nature14236.pdf

Non-Convex Optimization

Recall: Hidden Markov Models



$$P(x,y) = P(End \mid y^M)\prod_{i=1}^{M} P(y^i \mid y^{i-1})\prod_{i=1}^{M} P(x^i \mid y^i)$$

Recall: EM Algorithm for HMMs

- If we had y's ➔ max likelihood.
- If we had (A,O) ➔ predict y's

Chicken vs Egg!

1. Initialize A and O arbitrarily

Expectation Step

2. Predict prob. of y's for each training x

3. Use y's to estimate new (A,O)

Maximization Step

4. Repeat back to Step 1 until convergence

http://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm

Non-Convex Optimization Problem!

Converges to local optimum.

- If we had y’s ➔ max likelihood.
- If we had (A,O) ➔ predict y’s

Chicken vs Egg!

1. Initialize A and O arbitrarily

Expectation Step

2

3

4

Can We Train HMMs Optimally?

http://en.wikipedia.org/wiki/Baum%E2%80%93Welch_algorithm

Inspiration from Dimensionality Reduction

- Find best rank K approximation to Y:

$$\underset{U \in R^{NxK}, V \in R^{MxK}}{\operatorname{argmin}} \left\| Y - UV^T \right\|_2^2$$

- Non-convex optimization problem!
  - Due to non-convex feasible region

- **But optimally solved via SVD!**

Spectral Learning of HMMs

Want to Estimate:

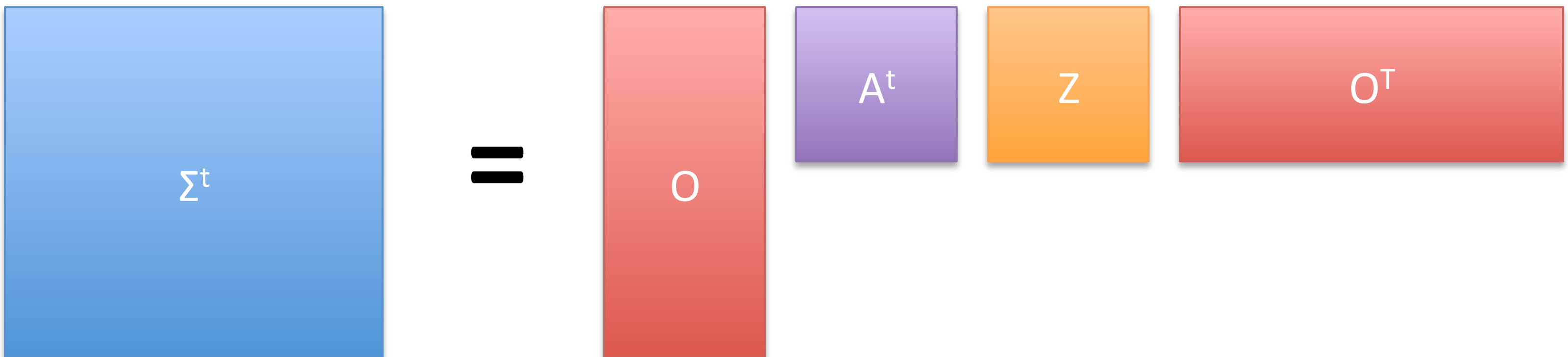
$$P(y^j \mid y^{j-1}) = A \qquad P(x^j \mid y^j) = O$$

$$\begin{aligned}
\Sigma^t = E\left[x^{j+t}\left(x^j\right)^T\right] &= E\left[E\left[x^{j+t}\left(x^j\right)^T \middle| y^j\right]\right] \\
&= E\left[E\left[x^{j+t} \middle| y^j\right]E\left[\left(x^j\right)^T \middle| y^j\right]\right] \\
&= E\left[\left(OA^t k y^j\right)\left(Oy^j\right)^T\right] \\
&= OA^t E\left[y^j\left(y^j\right)^T\right]O^T \\
&= OA^t Z O^T
\end{aligned}$$

Treat each x^j and y^j as indicator vector

http://www.cs.cmu.edu/~ggordon/spectral-learning/

Spectral Learning of HMMs



Rank-K SVD of Σ^1

Optimal Solution: $$A = U^T \Sigma^2 \left(U^T \Sigma^1\right)^{-1}$$

(requires a lot of data)

http://www.cs.cmu.edu/~ggordon/spectral-learning/

…and many more topics!

- Probabilistic Models
- Representation Learning
  - Deep learning is the most visible example
- Causal Reasoning
- ML + Game Theory
- ML + Systems
  - Large Scale Machine Learning
- Etc …

CS 159

- Special Topics in Machine Learning
  - Taught Every Spring Term
  - Topics Rotate
- **Next Term:**
  - "Online Learning, Interactive Machine Learning, and Learning from Human Feedback"
- Paper Reading & Presenting + Final Project
  - Graded on participation and final project