Announcements

- Posters CSE Atrium Thursday 10-12:30 (w/ coffee, bagels)
  - Turn in digital copy of poster (in PDF) on canvas (one for each student)
  - Prepare 1 minute speech for poster (we walk at 2 min)
    - Problem you are solving
    - Data you used
    - ML methodology and metrics used for evaluation
    - Results
  - We provide poster board and pins
  - Both one large poster (recommended) and several pinned pages are OK
  - If you didn't see us, you didn't get a grade.

Active Learning, classification

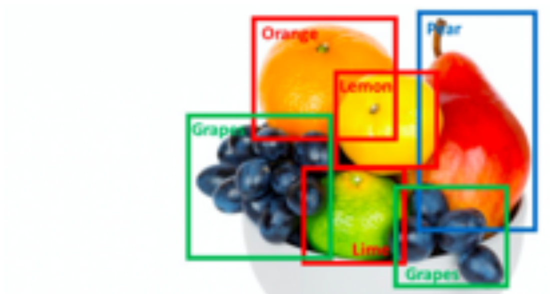
Machine Learning – CSE4546

Kevin Jamieson

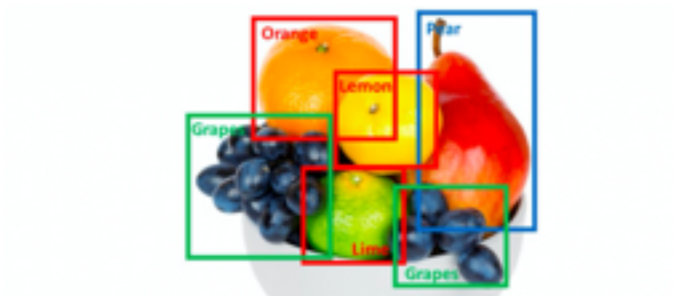
University of Washington

December 5, 2017

Impressive recent advances in image recognition and translation…

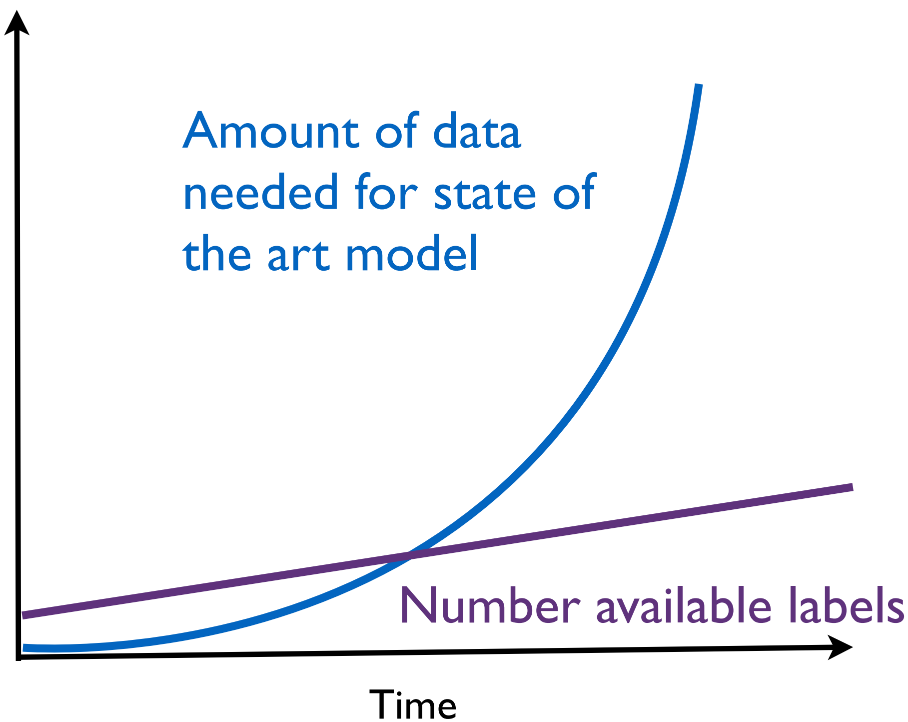


Impressive recent advances in image recognition and translation…



Google Translate

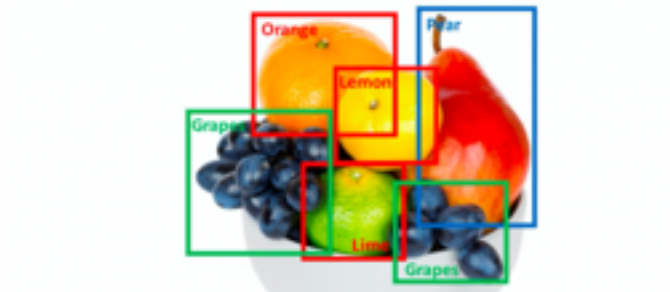
Break through language barriers.

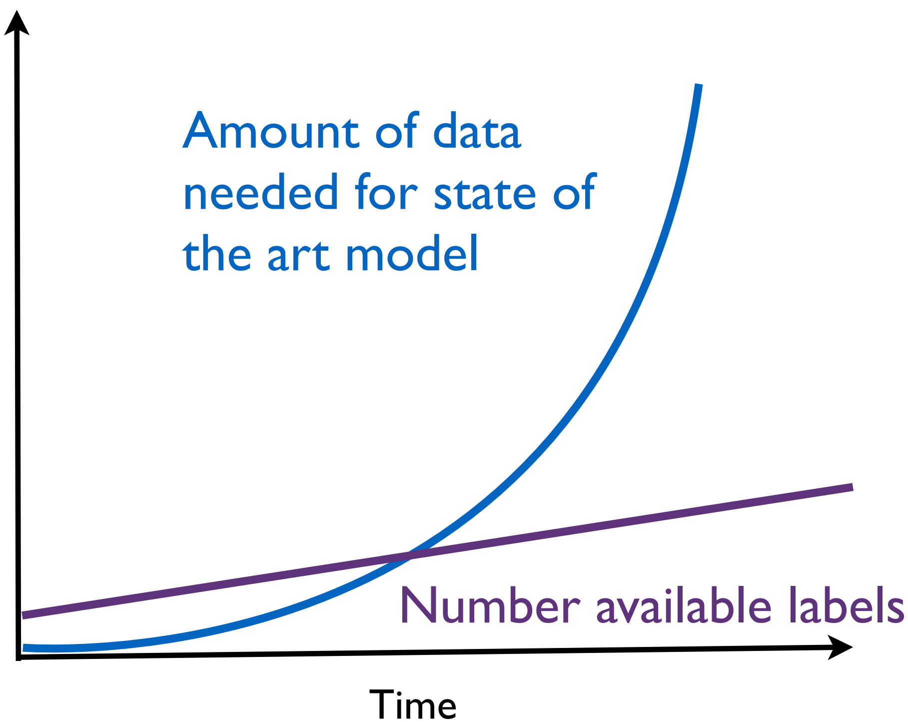


Challenges for large models:

1) An enormous amount of ***labeled data*** is necessary for training

Impressive recent advances in image recognition and translation…



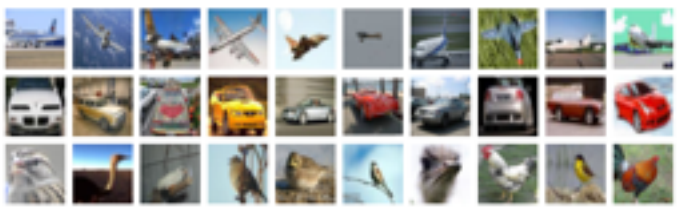


Challenges for large models:

1) An enormous amount of ***labeled data*** is necessary for training

2) An enormous amount of ***wall-clock time*** is necessary for training

Example: Image recognition



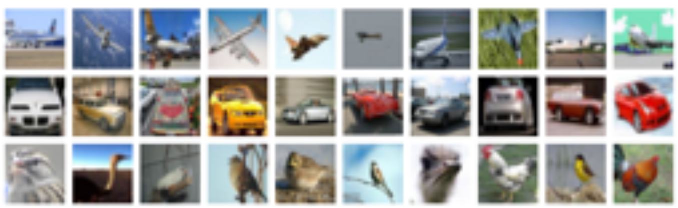
airplane

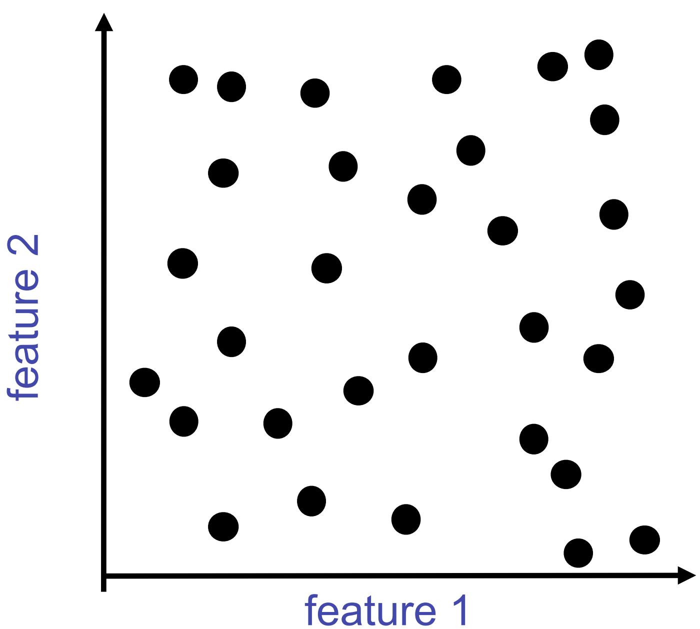
automobile

bird

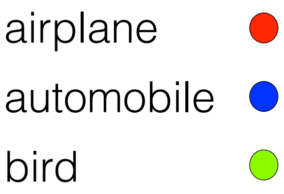
Example: Image recognition

airplane
automobile
bird

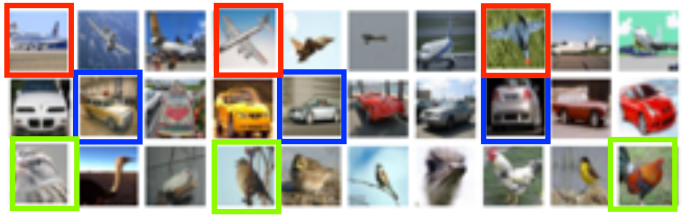


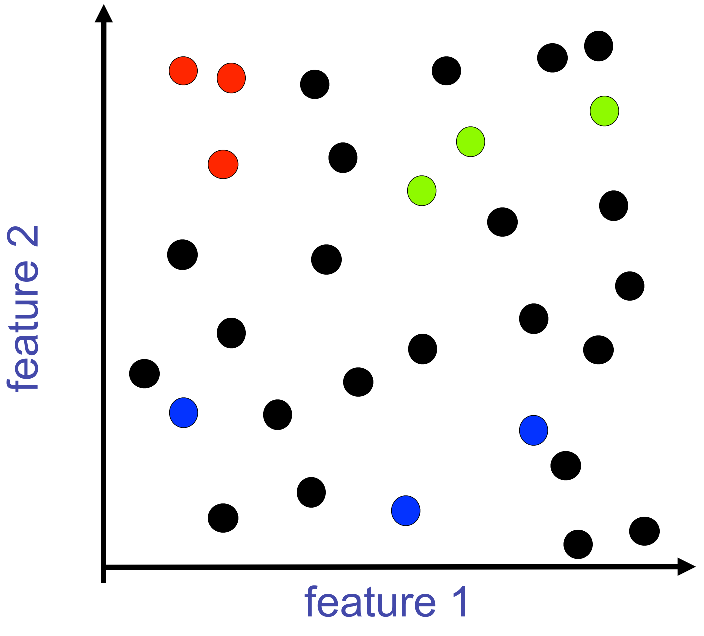


Example: Image recognition



Nonadaptive label assignment

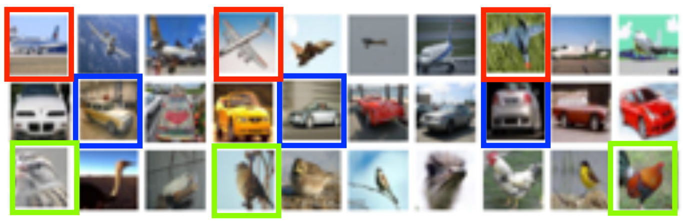


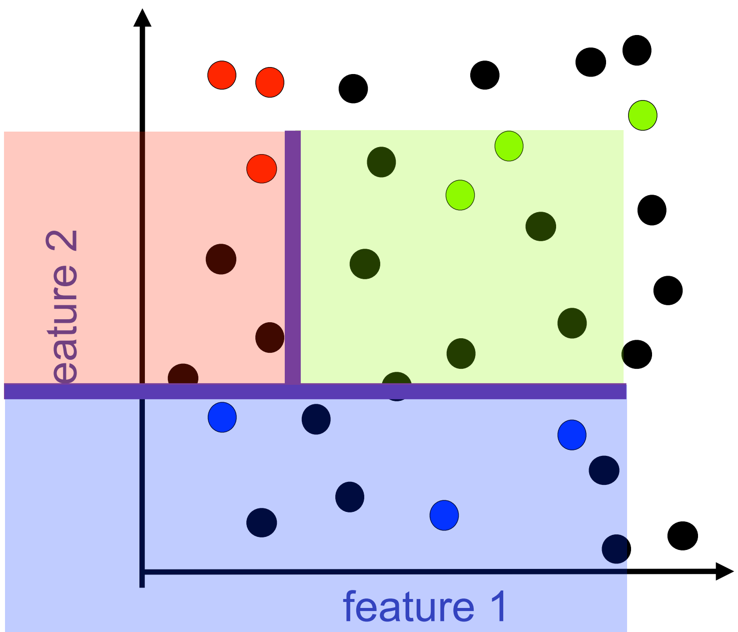


Example: Image recognition

airplane
automobile
bird

Nonadaptive label assignment

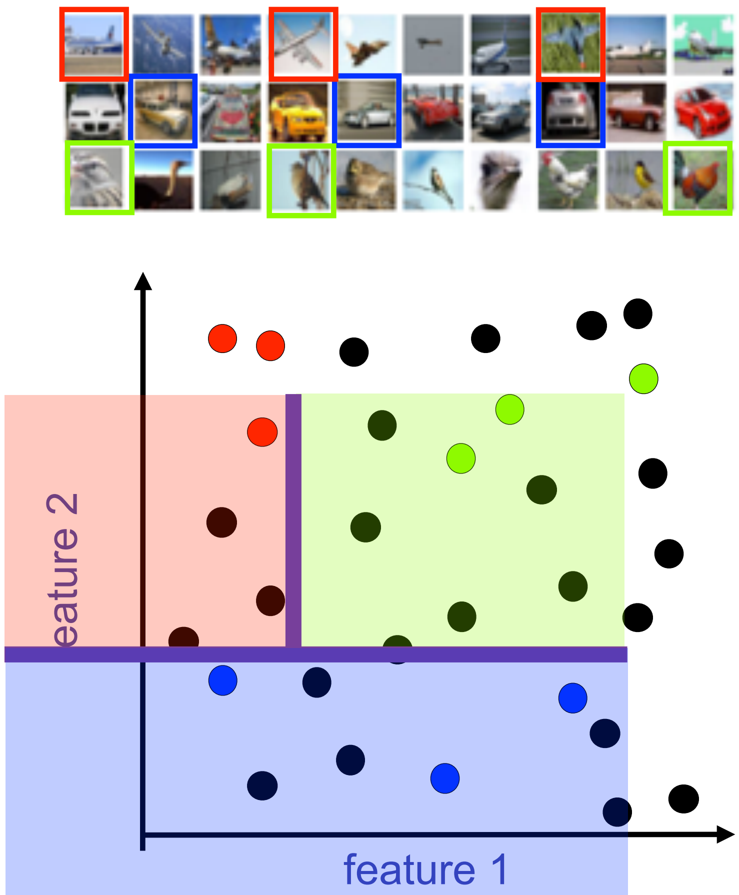




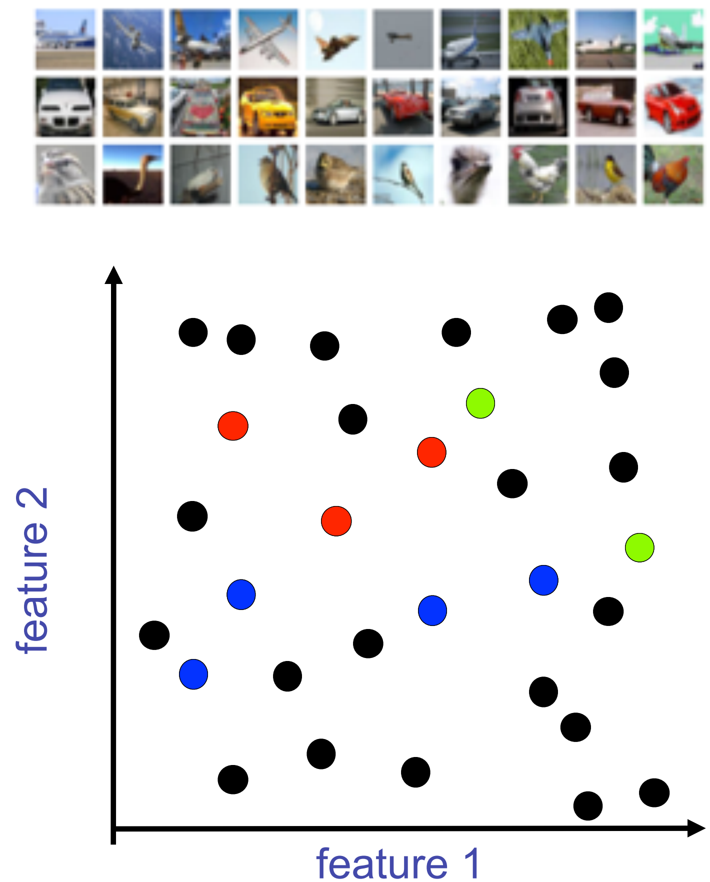
Example: Image recognition

airplane
automobile
bird

Nonadaptive label assignment



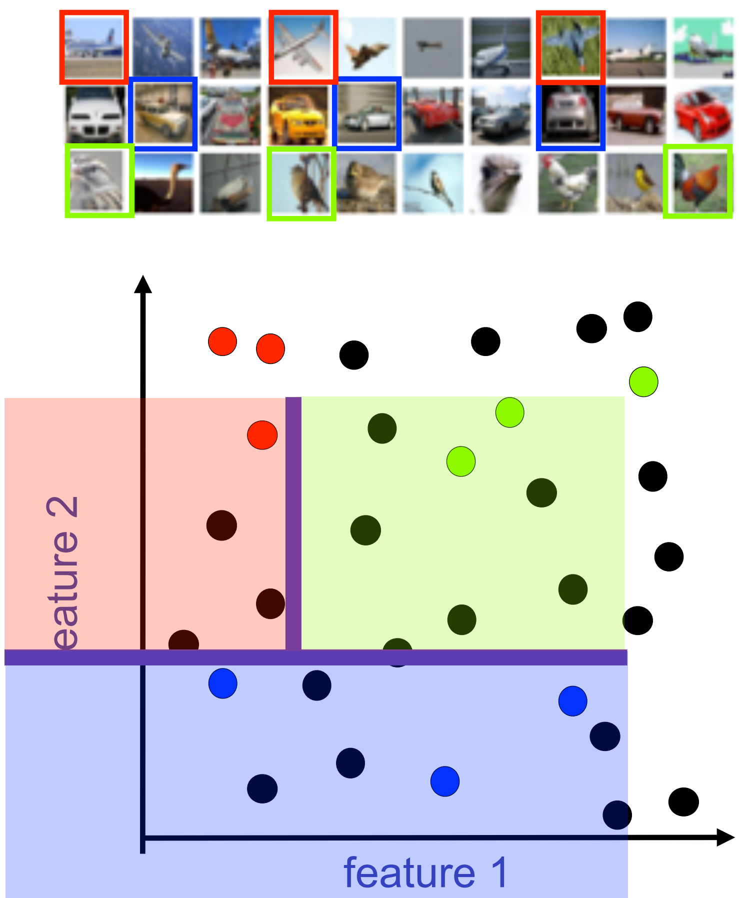
Adaptive label assignment



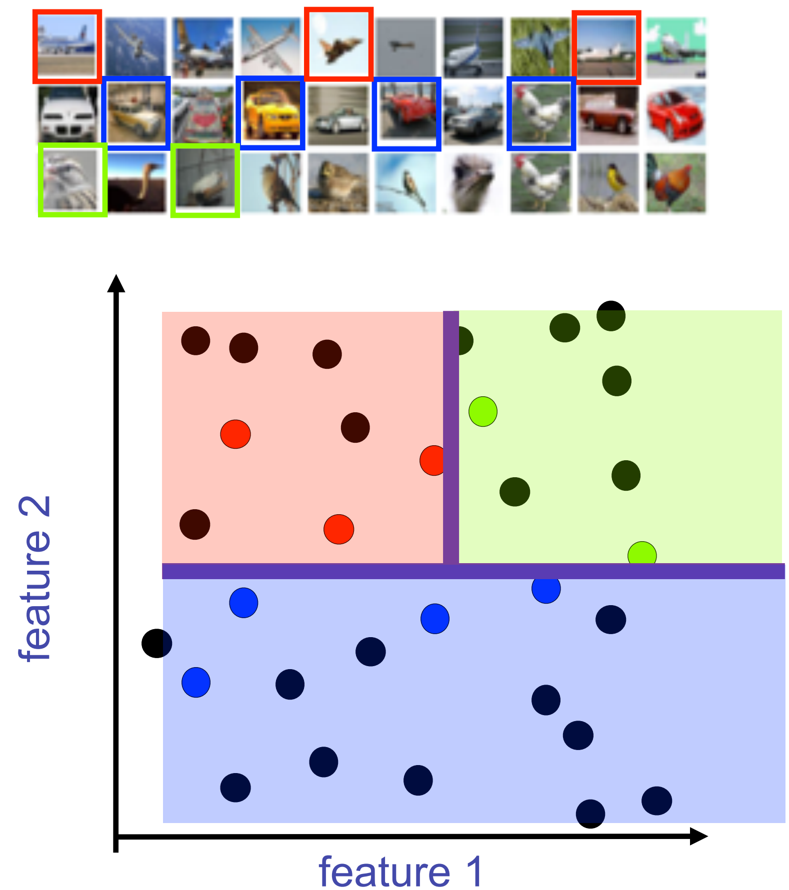
Example: Image recognition

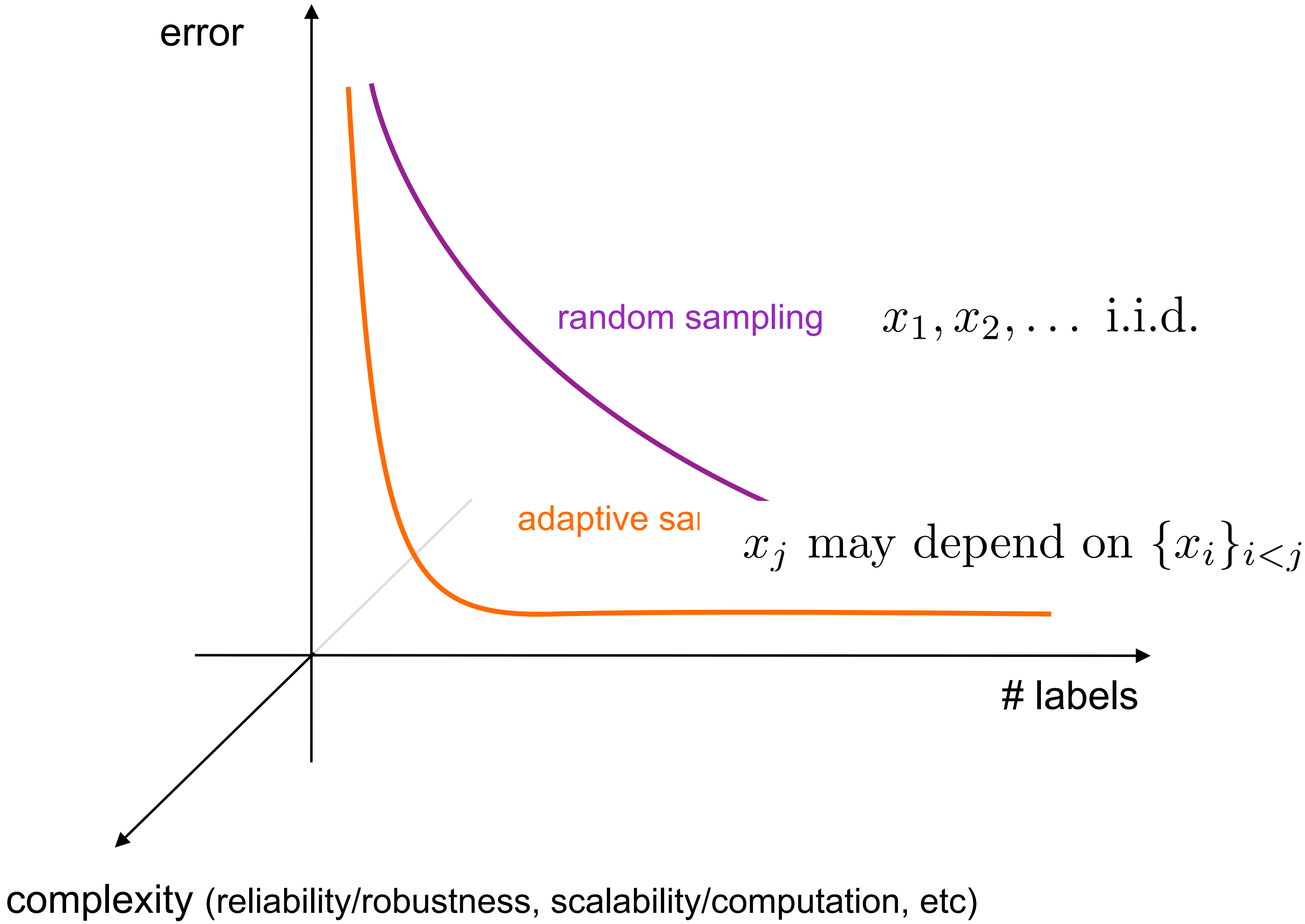
airplane
automobile
bird

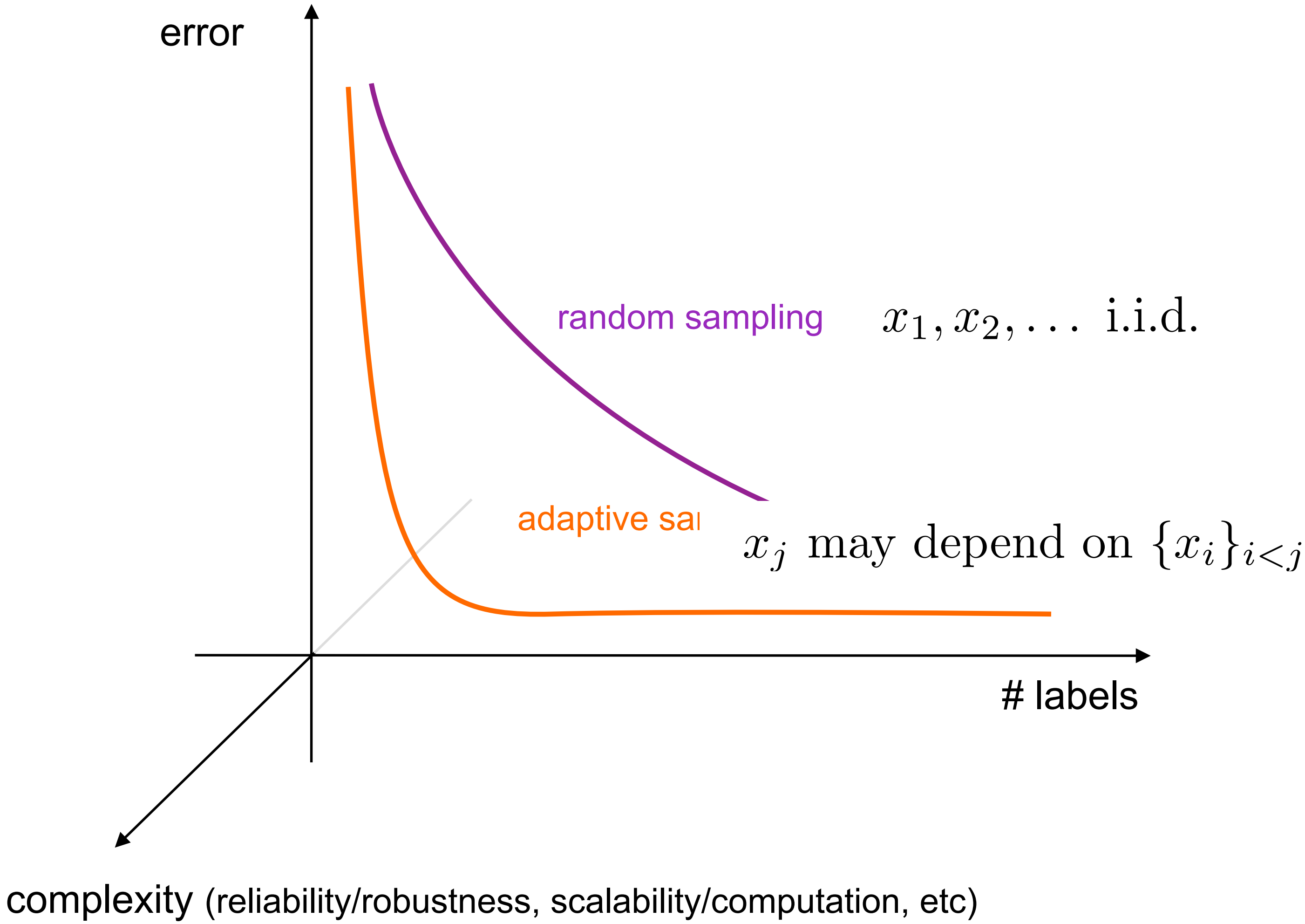
Nonadaptive label assignment



Adaptive label assignment




error
random sampling
$x_1, x_2, \ldots$ i.i.d.
adaptive sa
x_j may depend on $\{x_i\}_{i<j}$
labels
complexity (reliability/robustness, scalability/computation, etc)



Being convinced that data-collection ***should be adaptive*** is not the same thing as knowing ***how to be adaptive***.

THE NEW YORKER
CARTOON CAPTION CONTEST

Caption Contest #553
January 20, 2017

Third *“Maybe his second week will go better”*

Second *“I’d like to see other people”*

First *“The corrupt media will blow this way out of proportion”*

THE NEW YORKER CARTOON CAPTION CONTEST

Bob Mankoff
Cartoon Editor, The New Yorker

- $n \approx 5000$ captions submitted each week
- crowdsource contest to volunteers who rate captions
- goal: identify funniest caption

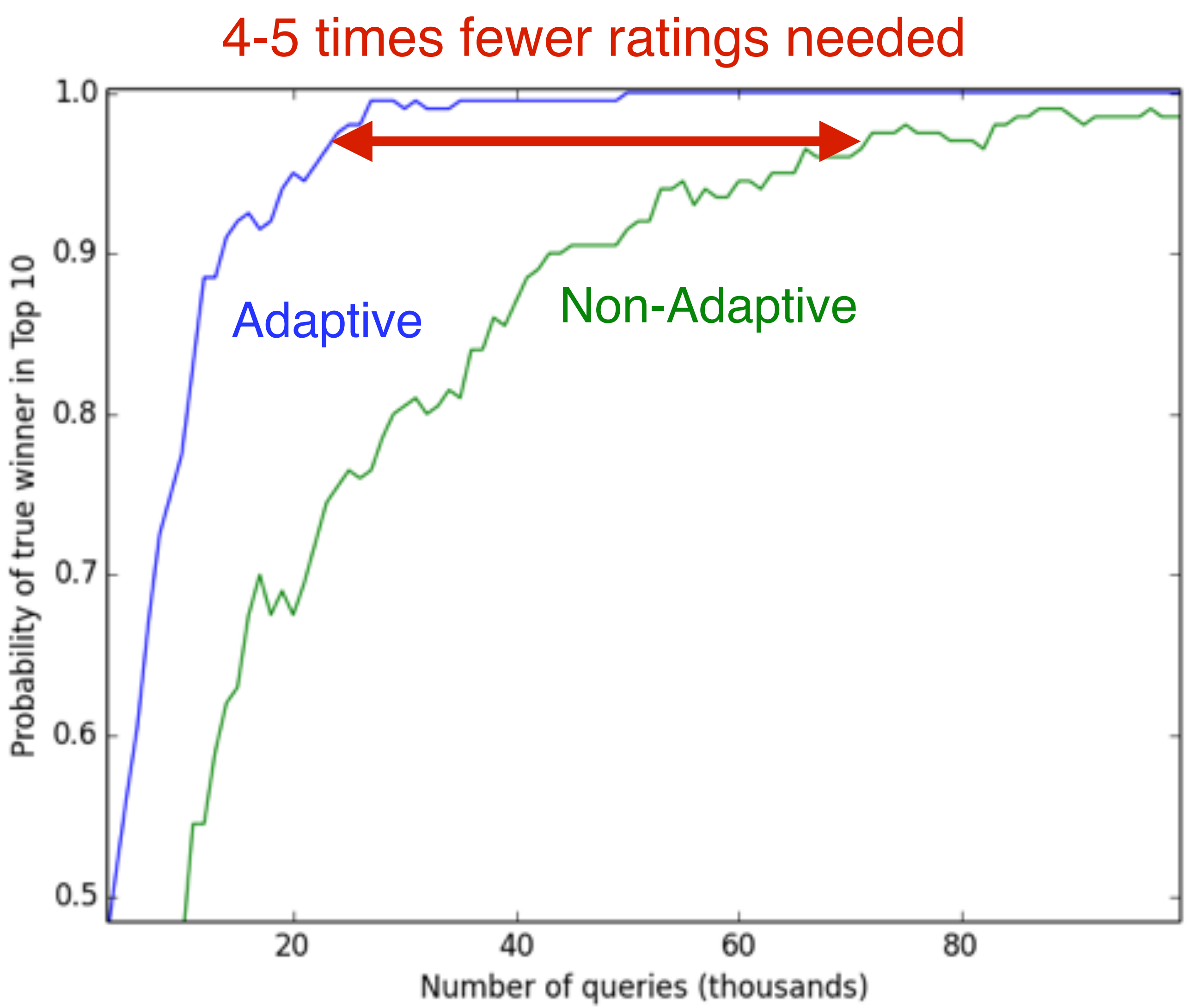
newyorker.com/cartoons/vote


Vote - The New Yorker
Kevin
www.newyorker.com/cartoons/vote
NEW
THE NEW YORKER
“It's amazing to think he started out in the lobby.”
UNFUNNY
FUNNY
DONE


Vote - The New Yorker
Kevin
www.newyorker.com/cartoons/vote
THE NEW YORKER
“I thought all our plants moved to Mexico.”
UNFUNNY
FUNNY
DONE

Which caption do we show next?

1) Non-adaptive uniform distribution over captions (A/B testing)
2) Adaptive: stop showing captions that will not win

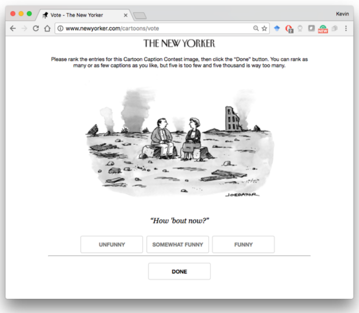


Which caption do we show next?

1) Non-adaptive uniform distribution over captions (A/B testing)
2) Adaptive: stop showing captions that will not win

Best-action identification problem

Stopping rule



While algorithm does not exit:

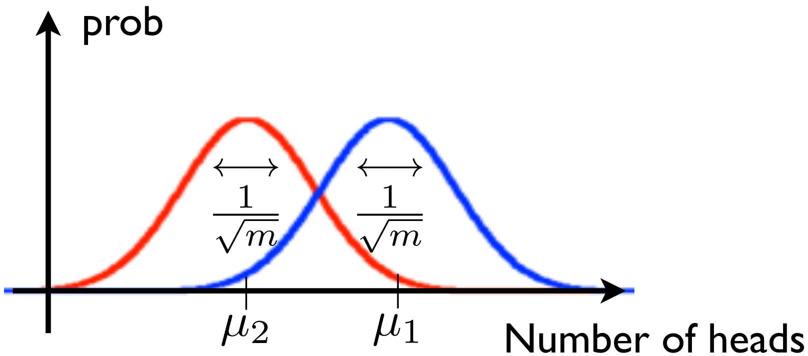
- algorithm shows caption $i \in \{1, \ldots, n\}$
- Observe iid Bernoulli with $\mathbb{P}(\text{“funny”}) = \mu_i$

Sampling rule

Objective: with probability .99, identify $\arg\max_{i=1,\ldots,n} \mu_i$ using as few total samples as possible

Best-arm Identification n=2

Consider $n = 2$ and flip coins $i = 1, 2$ to get $X_{i,1}, X_{i,2}, \ldots, X_{i,m}$



$$\widehat{\mu}_{i,m} = \frac{1}{m} \sum_{j=1}^{m} X_{i,j}$$

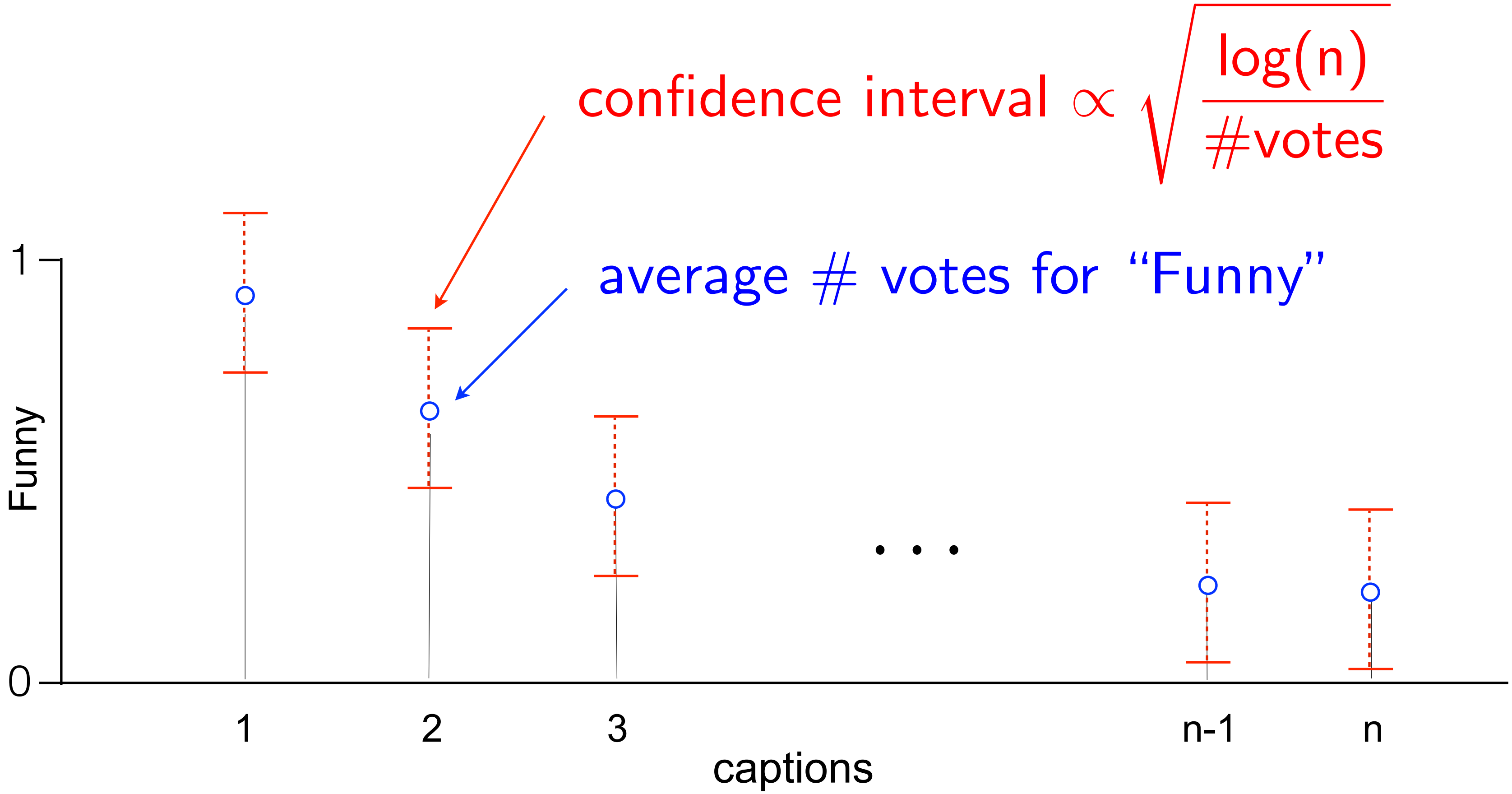
Test: $\widehat{\mu}_{1,m} - \widehat{\mu}_{2,m} \geq 0$

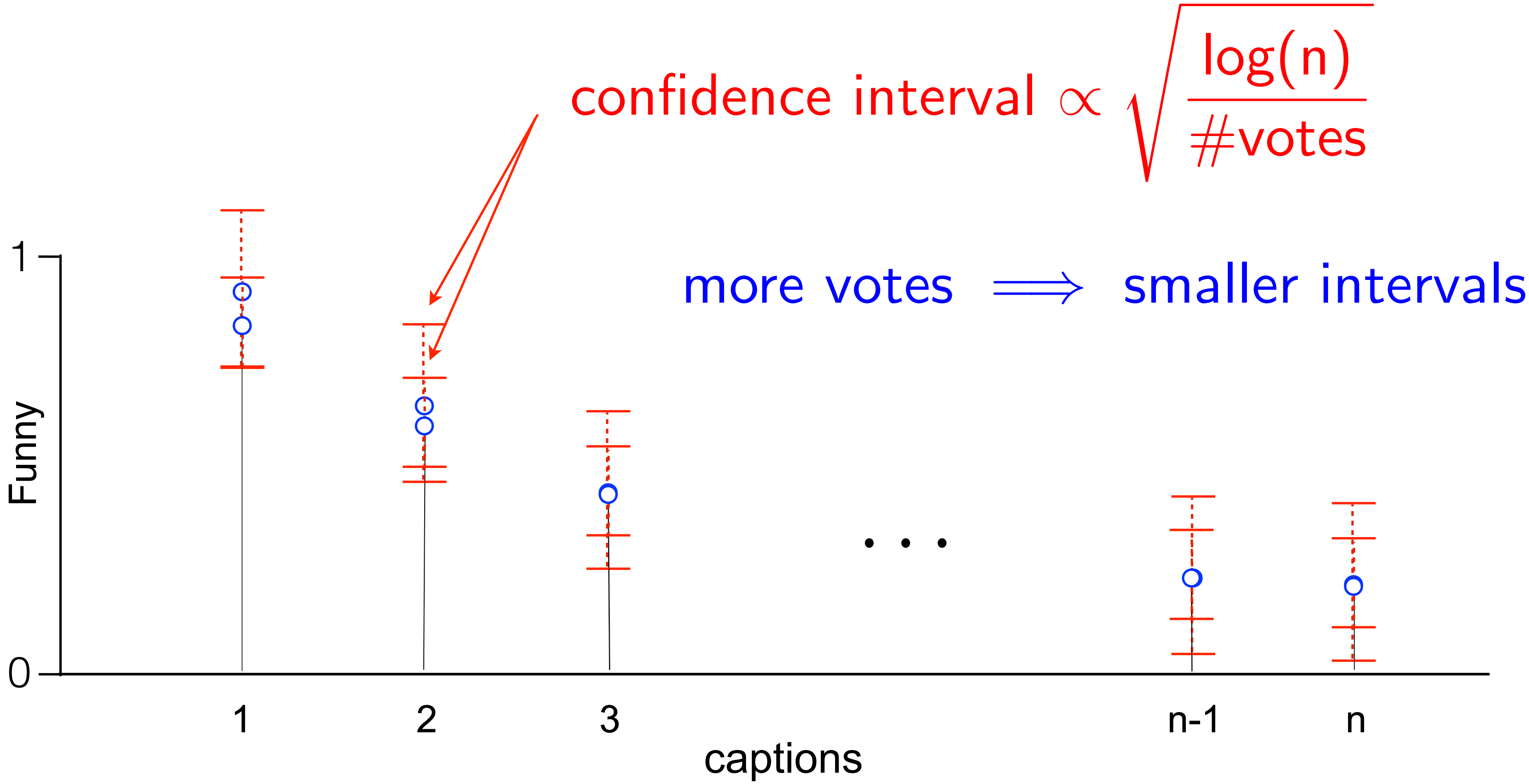
By a Chernoff Bound, if $\Delta = \mu_1 - \mu_2$ then

$$m = 2\log(1/\delta)\Delta^{-2} \implies \widehat{\mu}_{1,m} > \widehat{\mu}_{2,m} + 2\sqrt{\frac{\log(1/\delta)}{2m}} \implies \mu_1 > \mu_2$$

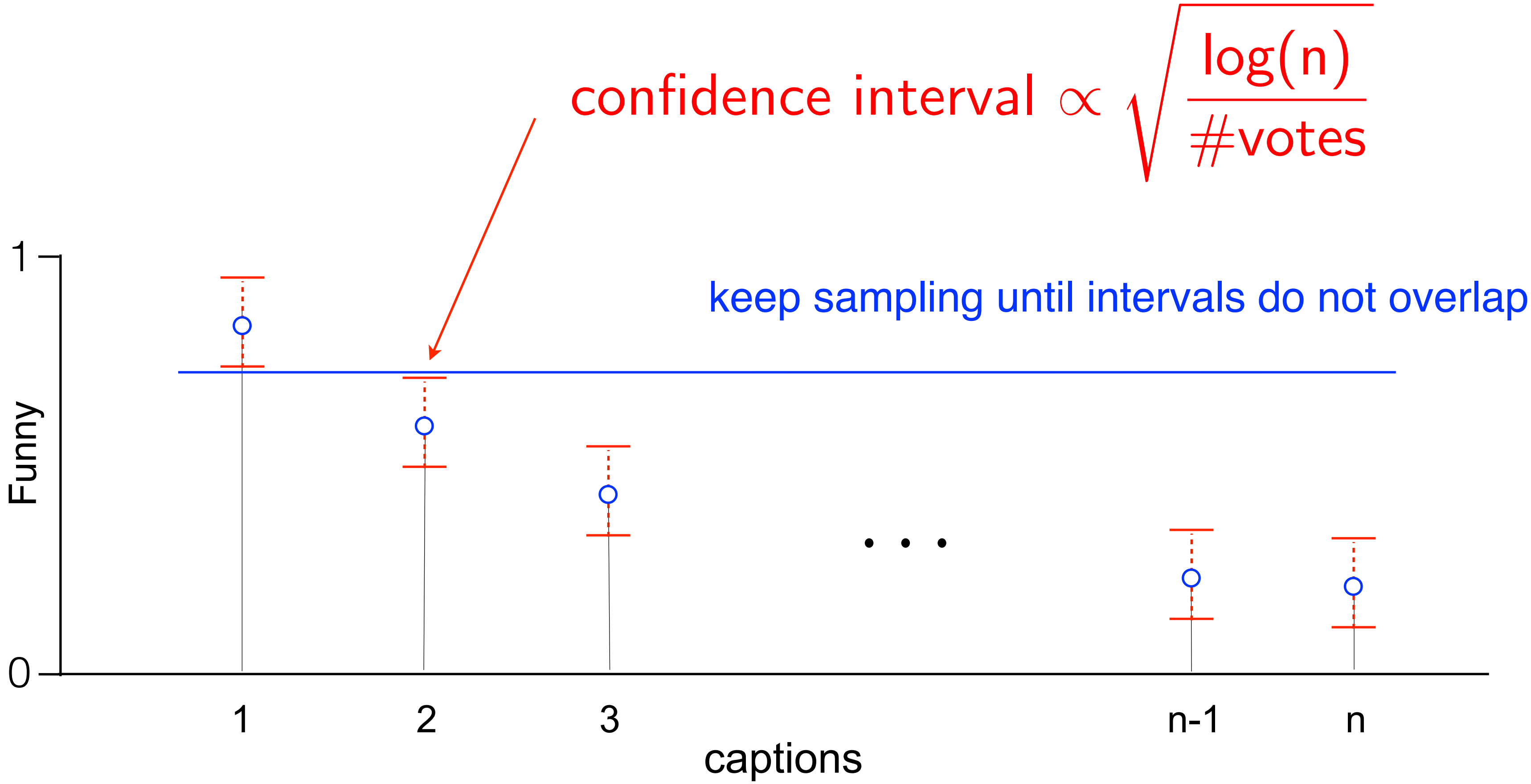
with probability $\geq 1 - 2\delta$

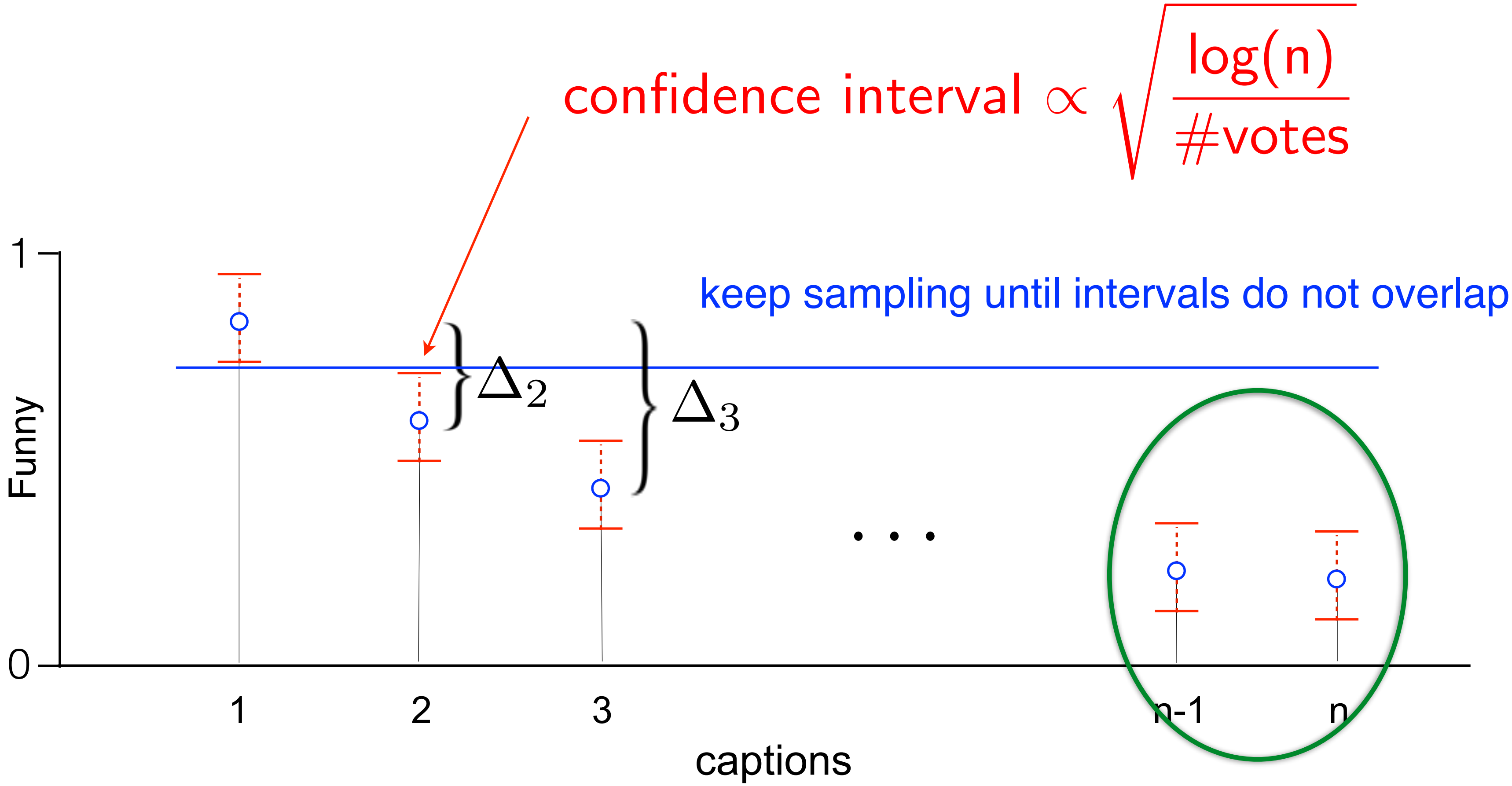
Arm 1 lower confidence bound > Arm 2 upper confidence bound


confidence interval $\propto \sqrt{\frac{\log(n)}{\#\text{votes}}}$
average # votes for "Funny"
Funny
1
0
1
2
3
• • •
n-1
n
captions



confidence interval $\propto \sqrt{\frac{\log(n)}{\#\text{votes}}}$
more votes $\Longrightarrow$ smaller intervals
1
0
Funny
1
2
3
n-1
n
captions


confidence interval $\propto \sqrt{\frac{\log(n)}{\#\text{votes}}}$
keep sampling until intervals do not overlap
1
0
Funny
1
2
3
n-1
n
captions



votes Non-adaptive: $n \max_{i=1,\ldots,n} \Delta_i^{-2} \log(n)$

Successive Elimination [Even-dar…'06]: $\sum_{i=1}^{n} \Delta_i^{-2} \log(n)$

Stop sampling caption *i* as soon as no overlap

But this treated all captions the same, ignoring the text of the caption! Last lecture we talked about **extracting features from documents and text to represent them as vectors** (e.g., tf*idf or word2vec)

"The corrupt media will blow this way out of proportion" —feature extraction→ $x \in \mathbb{R}^d$

Using features, instead of finding the *crowd's* favorite caption, let's find ***your*** favorite caption!

Better yet, instead of captions, what about beer?

“Find my beer” problem

Bartender: “Try these samples. Was it closer to A or B?”

Me: “A”

Bartender: “Ok, try these, C or D?”

Me: “D”

⋮

Me: “You found it!”

Beer has some “simple” description and he was adaptively choosing questions to explore this space, **much like 20 questions**.

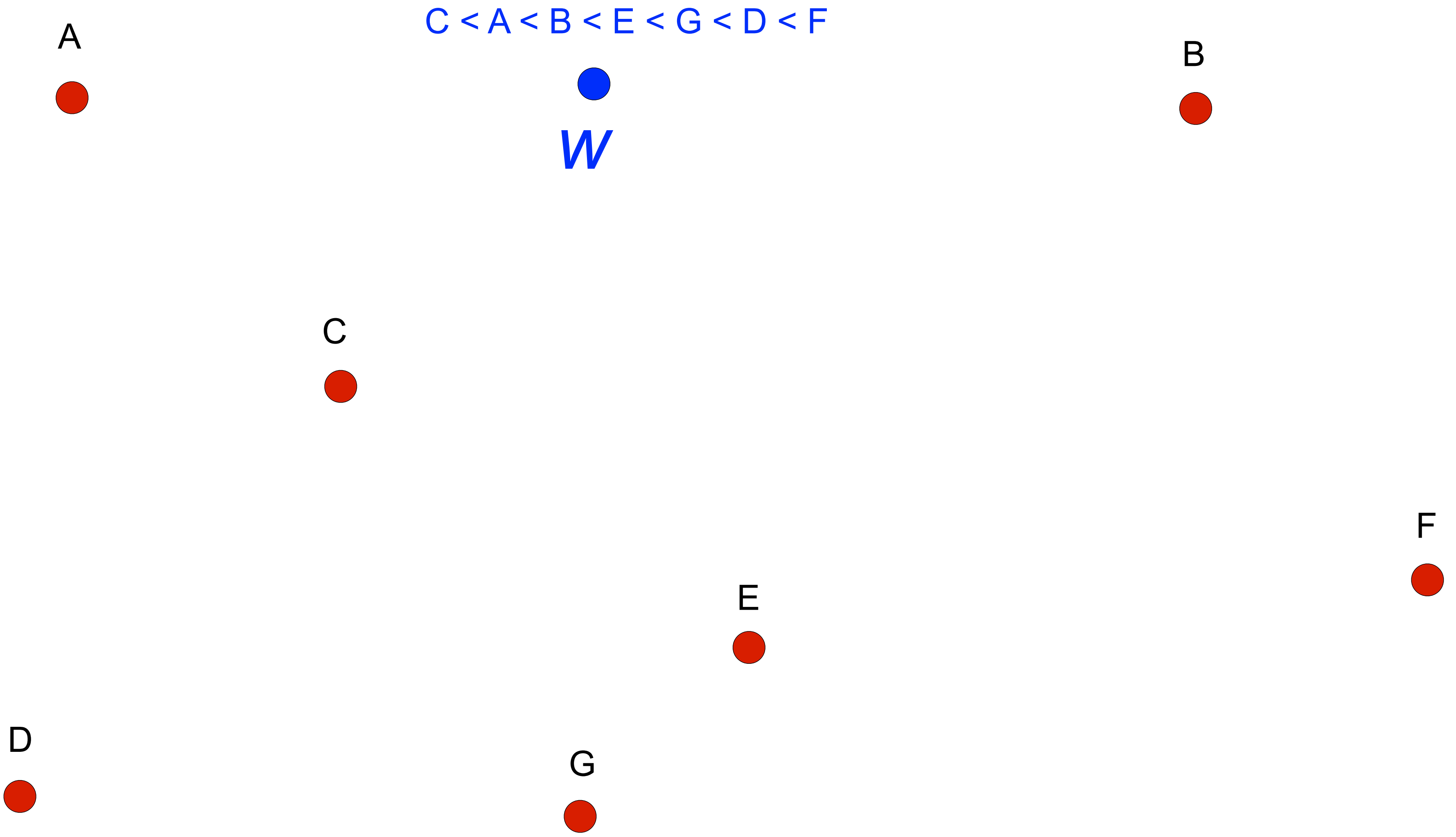
Optimization

Consider n beers, each described by a d-dimensional feature vector: $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$.

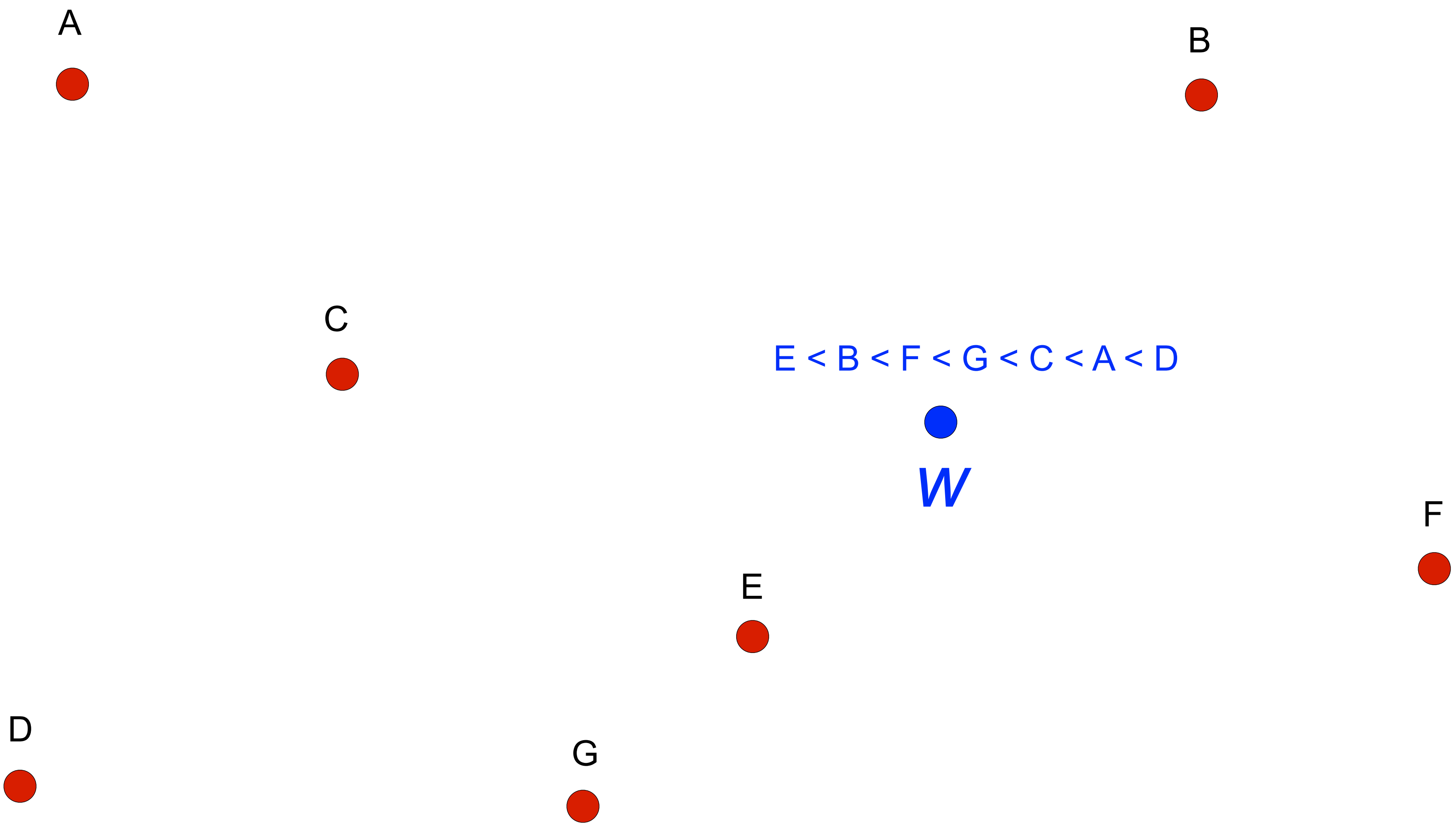
Goal: Learn someone's preferences over the n objects under the *ideal point* model:

$$\forall (i,j), \text{ object } i \text{ is preferred to } j \iff ||w - x_i||_2 < ||w - x_j||_2 \text{ for some } w \in \mathbb{R}^d$$

Ranking According to Distance

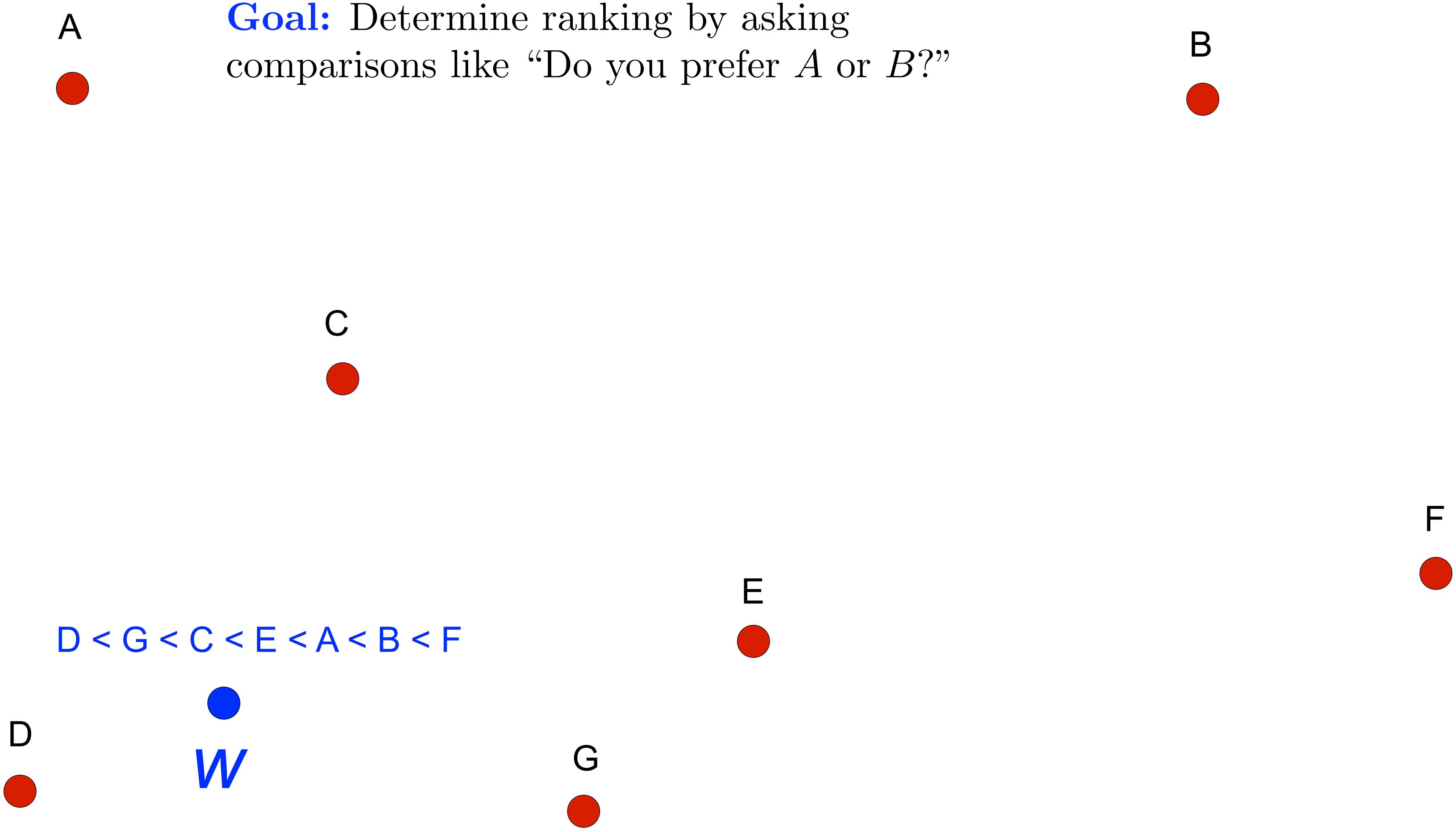


Ranking According to Distance



Ranking According to Distance

Goal: Determine ranking by asking comparisons like "Do you prefer A or B?"



Optimization

Consider n beers, each described by a d-dimensional feature vector: $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$.

Goal: Learn someone's preferences over the n objects under the *ideal point* model:

$$\forall (i,j), \text{ object } i \text{ is preferred to } j \iff ||w - x_i||_2 < ||w - x_j||_2 \text{ for some } w \in \mathbb{R}^d$$

binary information we can gather: $q_{i,j} \equiv$ **do you prefer** x_i **or** x_j

CAL algorithm principle introduced in: D Cohn, L Atlas, and R Ladner, "Improving generalization with active learning," Machine learning 15 (2), pp. 201-221, 1994.

Lazy Binary Search

input: $x_1, \ldots, x_n \in \mathbb{R}^d$
initialize: $x_1, \ldots, x_n$ in uniformly random order

for k=2,...,n
  for i=1,...,k-1
    if $q_{i,k}$ is ***ambiguous*** given $\{q_{i,j}\}_{i,j<k}$,
      then ask for pairwise comparison,
    else impute $q_{i,j}$ from $\{q_{i,j}\}_{i,j<k}$

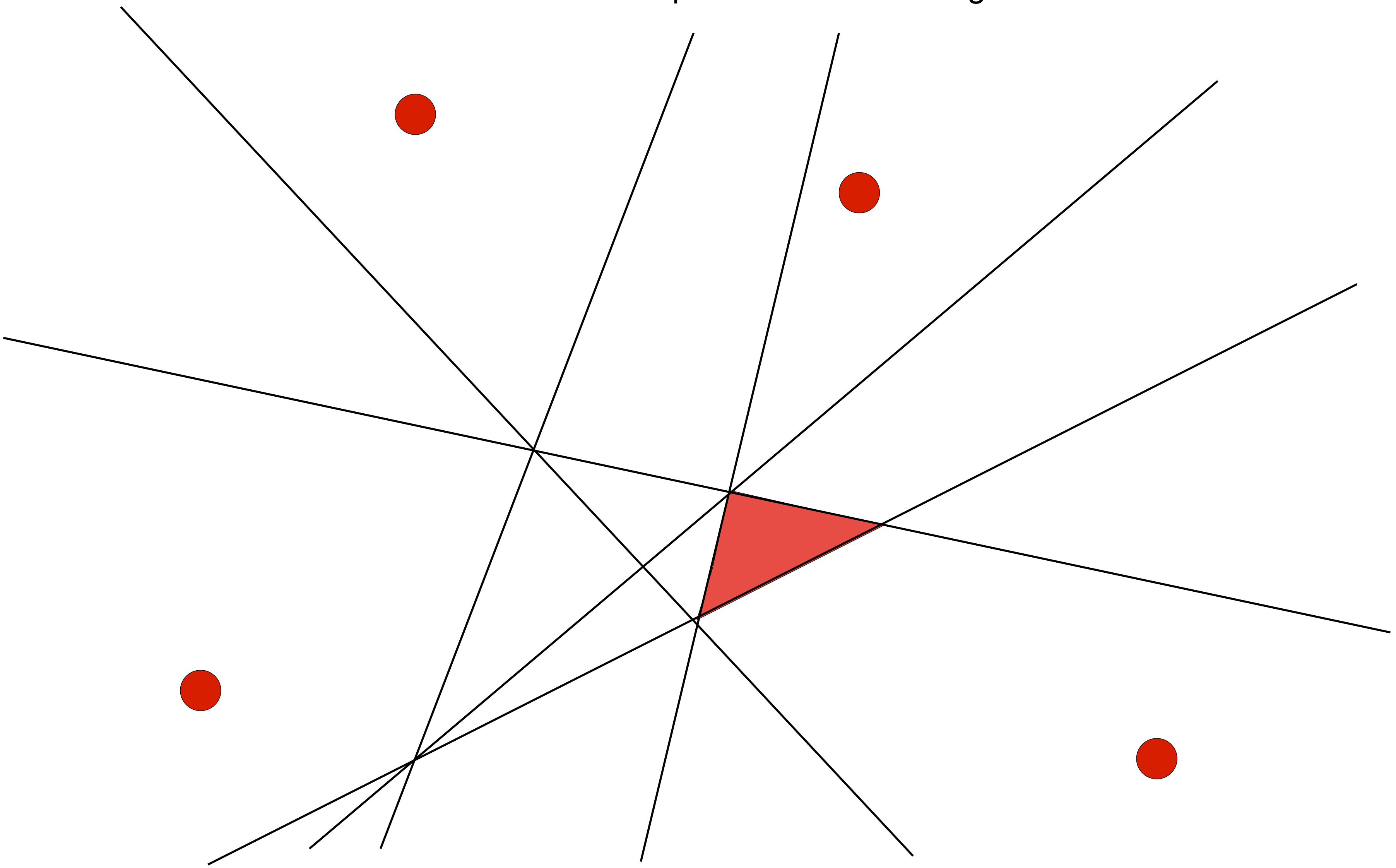
output: ranking of $x_1, \ldots, x_n$ consistent with *all* pairwise comparisons

simple linear program

Ranking and Geometry

suppose we have ranked 4 beers

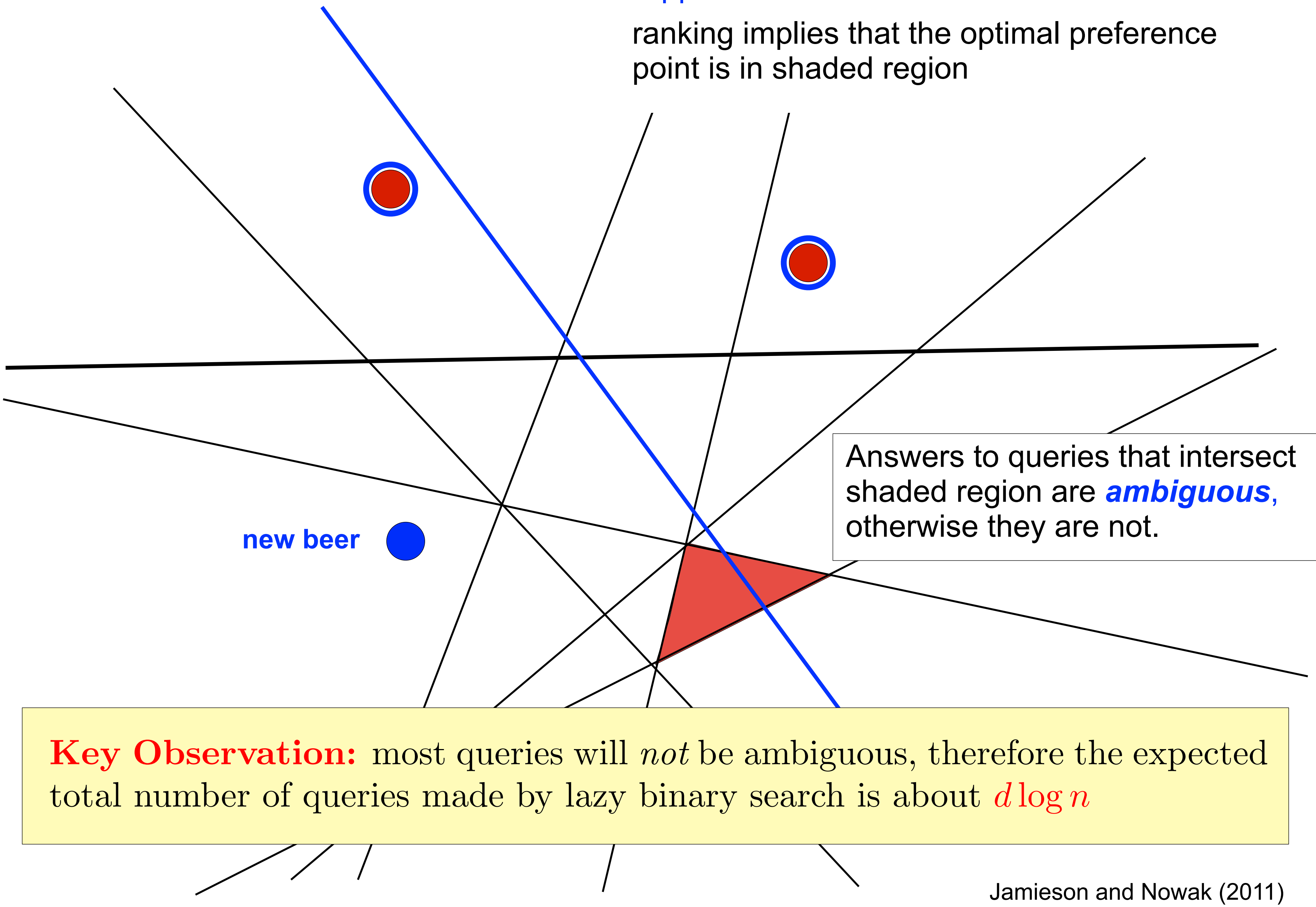
ranking implies that the optimal preference point is in shaded region



Ranking and Geometry

suppose we have ranked 4 beers

ranking implies that the optimal preference point is in shaded region



Answers to queries that intersect shaded region are ***ambiguous***, otherwise they are not.

Key Observation: most queries will *not* be ambiguous, therefore the expected total number of queries made by lazy binary search is about $d \log n$

Jamieson and Nowak (2011)

Ranking and Geometry

at k-th step of algorithm

$$\# \text{ of } d\text{-cells} \approx \frac{k^{2d}}{d!} \qquad \text{(Coombs 1960)}$$

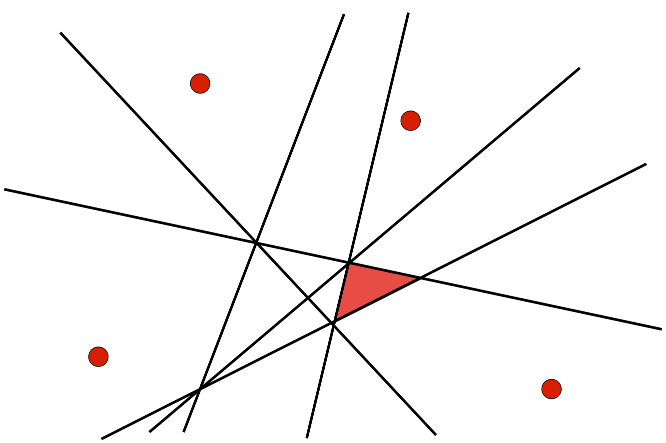
$$\# \text{ intersected} \approx \frac{k^{2(d-1)}}{(d-1)!} \qquad \text{(Buck 1943)}$$

$$\implies \mathbb{P}(\text{ambiguous}) \approx \frac{d}{k^2} \qquad \text{(Cover 1965)}$$

$$\implies \mathbb{E}[\#\text{ambiguous}] \approx \frac{d}{k}$$

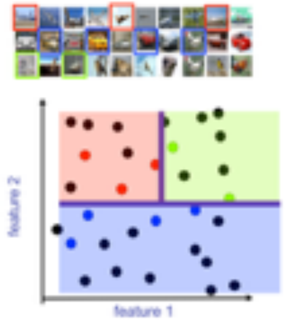
$$\implies \mathbb{E}[\# \text{ requested}] \approx \sum_{k=2}^{n} \frac{d}{k} \qquad \text{(Jamieson \& Nowak 2011)}$$

$$\approx d \log n$$



Commercial application:
Amazon visual search:
https://shopbylook.amazon.com/

Classification with adaptively collected dataset



“Find the best” Judgments from a crowd
(and adaptive A/B testing)

“Find the best” with features

Pure Exploration

Find and use ad with highest click-through-rate

Balance of **exploration versus exploitation**

Reinforcement Learning & Markov Decision Processes (MDPs)

Machine Learning – CSE546

Kevin Jamieson
University of Washington

December 5, 2017

Learning to act

- Reinforcement learning
- An agent
  - Makes sensor observations
  - Must select action
  - Receives rewards
    - positive for "good" states
    - negative for "bad" states

[Ng et al. '05]

Markov Decision Process (MDP) Representation

- State space:
  - Joint state **x** of entire system
- Action space:
  - Joint action **a**= $\{a_1, \ldots, a_n\}$ for all agents
- Reward function:
  - Total reward R(**x**,**a**)
    - sometimes reward can depend on action
- Transition model:
  - Dynamics of the entire system P(**x**'|**x**,**a**)

Discount Factors

People in economics and probabilistic decision-making do this all the time.

The "Discounted sum of future rewards" using discount factor γ" is

(reward now) +

γ (reward in 1 time step) +

γ^2 (reward in 2 time steps) +

γ^3 (reward in 3 time steps) +

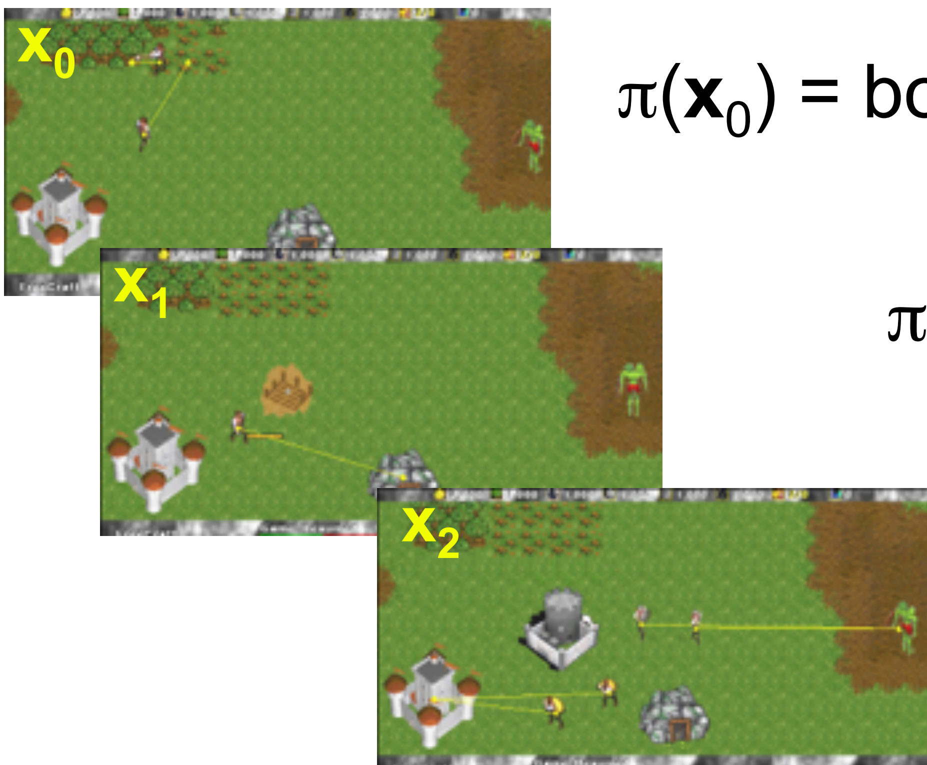
:

: (infinite sum)

Policy

Policy: $\pi(\mathbf{x}) = \mathbf{a}$

At state **x**, action **a** for all agents



$\pi(\mathbf{x}_0)$ = both peasants get wood

$\pi(\mathbf{x}_1)$ = one peasant builds barrack, other gets gold

$\pi(\mathbf{x}_2)$ = peasants get gold, footmen attack

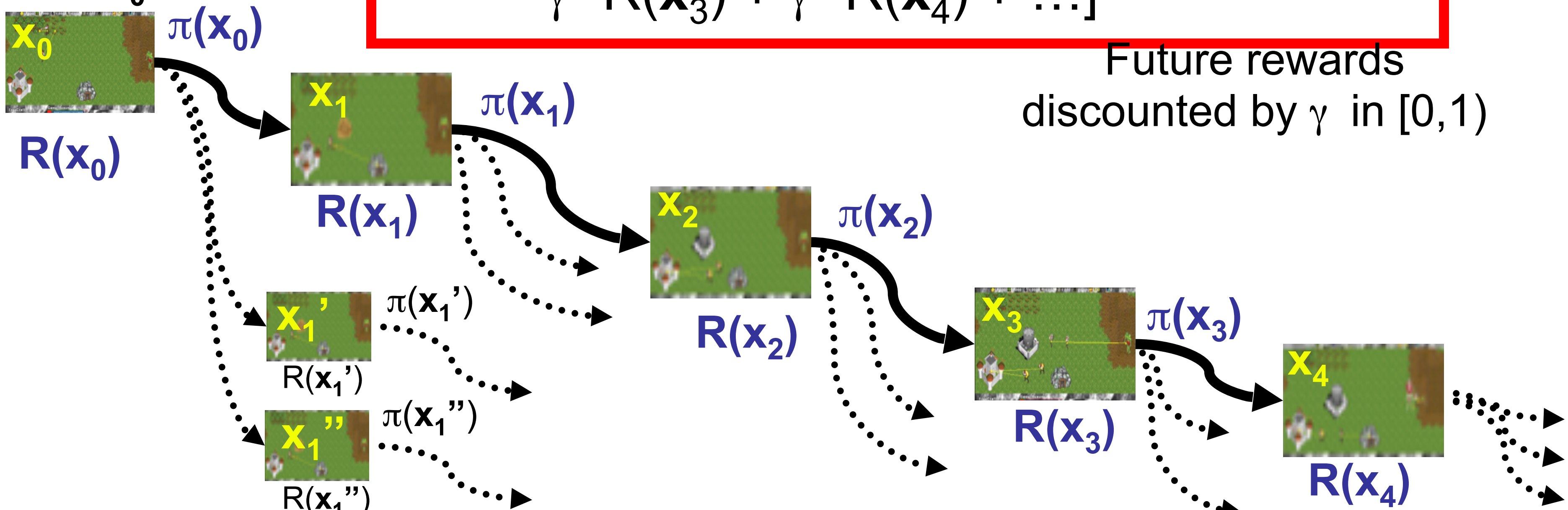
Value of Policy

Value: $V_\pi(\mathbf{x})$ → Expected long-term reward starting from **x**

$$V_\pi(\mathbf{x_0}) = \mathbf{E}_\pi[R(\mathbf{x}_0) + \gamma\, R(\mathbf{x}_1) + \gamma^2\, R(\mathbf{x}_2) + \gamma^3\, R(\mathbf{x}_3) + \gamma^4\, R(\mathbf{x}_4) + \ldots]$$

Future rewards discounted by γ in [0,1)

Start from $\mathbf{x_0}$



Computing the value of a policy

$V_\pi(\mathbf{x_0}) = \mathbf{E}_\pi[R(\mathbf{x}_0) + \gamma\, R(\mathbf{x}_1) + \gamma^2\, R(\mathbf{x}_2) + \gamma^3\, R(\mathbf{x}_3) + \gamma^4\, R(\mathbf{x}_4) + \ldots]$

- Discounted value of a state:
  - value of starting from x_0 and continuing with policy π from then on

$$
\begin{aligned}
V_\pi(x_0) &= E_\pi[R(x_0) + \gamma R(x_1) + \gamma^2 R(x_2) + \gamma^3 R(x_3) + \cdots] \\
&= E_\pi[\sum_{t=0}^{\infty} \gamma^t R(x_t)]
\end{aligned}
$$

- A recursion!

Simple approach for computing the value of a policy: Iteratively

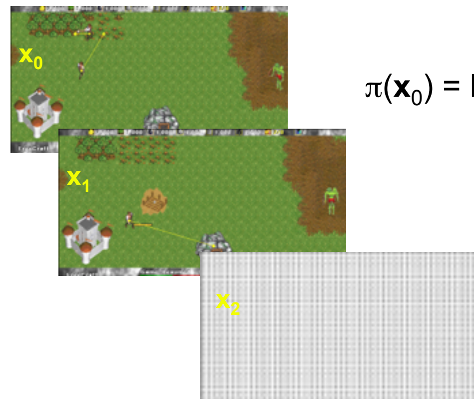
$$V_\pi(x) \;=\; R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi(x')$$

- Can solve using a simple convergent iterative approach: (a.k.a. dynamic programming)
  - Start with some guess V^0
  - Iteratively say:
    - $$V_\pi^{t+1}(x) \;\leftarrow\; R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi^t(x')$$
  - Stop when $||V_{t+1}-V_t||_\infty < \varepsilon$
    - means that $||V_\pi-V_{t+1}||_\infty < \varepsilon/(1-\gamma)$

But we want to learn a **Policy**

- So far, told you how good a policy is…
- But how can we choose the best policy???

Policy: $\pi(\mathbf{x}) = \mathbf{a}$ → At state $\mathbf{x}$, action $\mathbf{a}$ for all agents



$\pi(\mathbf{x}_0)$ = both peasants get wood

$\pi(\mathbf{x}_1)$ = one peasant builds barrack, other gets gold

$\pi(\mathbf{x}_2)$ = peasants get gold, footmen attack

- Suppose there was only one time step:
  - world is about to end!!!
  - select action that maximizes reward!

Unrolling the recursion

- Choose actions that lead to best value in the long run
  - Optimal value policy achieves optimal value V^*

$$V^*(x_0) = \max_{a_0} R(x_0, a_0) + \gamma E_{a_0}[\max_{a_1} R(x_1) + \gamma^2 E_{a_1}[\max_{a_2} R(x_2) + \cdots]]$$

Bellman equation

- Evaluating policy π:

$$V_\pi(x) \quad = \quad R(x) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V_\pi(x')$$

- Computing the optimal value V^* - Bellman equation

$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x}, \mathbf{a}) V^*(\mathbf{x}')$$

Optimal Long-term Plan

Optimal value function $V^*(\mathbf{x})$ → Optimal Policy: $\pi^*(\mathbf{x})$

Optimal policy:

$$\pi^*(\mathbf{x}) = \arg\max_a R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}')$$

Interesting fact – Unique value

$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}')$$

- *Slightly surprising fact*: There is only one V^* that solves Bellman equation!
  - there may be many optimal policies that achieve V^*
- *Surprising fact*: optimal policies are good everywhere!!!

$$V_{\pi^*}(x) \geq V_\pi(x), \ \ \forall x, \ \ \forall \pi$$

Solving an MDP

Solve Bellman equation → Optimal value $V^*(\mathbf{x})$ → Optimal policy $\pi^*(\mathbf{x})$

$$V^*(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^*(\mathbf{x}')$$

Many algorithms solve the Bellman equations:

- Policy iteration [Howard '60, Bellman '57]
- Value iteration [Bellman '57]
- Linear programming [Manne '60]
- …

Value iteration (a.k.a. dynamic programming) – t simplest of all

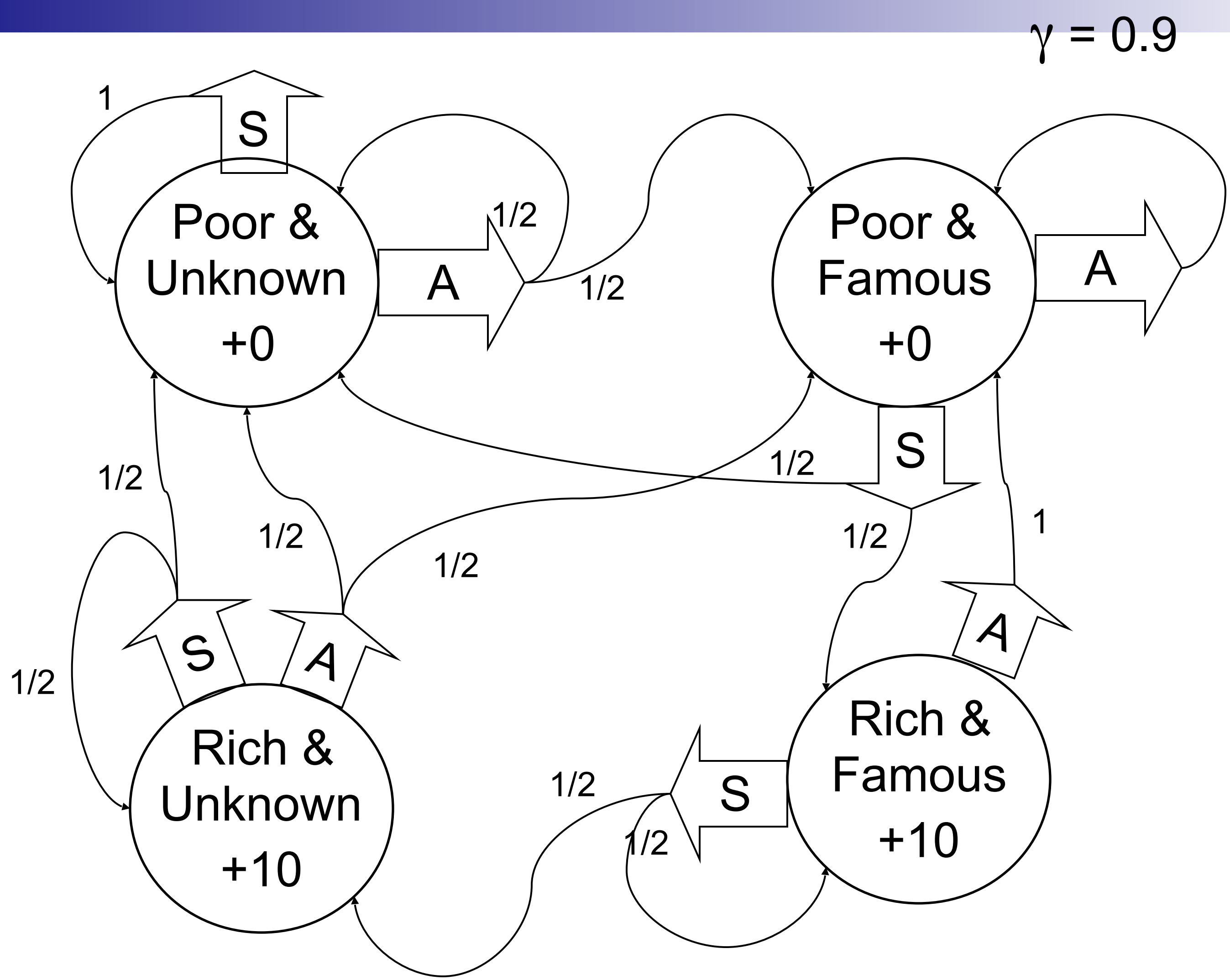
$$V^*(x) = R(x,a) + \gamma \sum_{x'} P(x' \mid x, a = \pi(x)) V^*(x')$$

- Start with some guess V^0
- Iteratively say:
  - $V^{t+1}(x) \leftarrow \max_a R(x,a) + \gamma \sum_{x'} P(x' \mid x, a) V^t(x')$
- Stop when $||V_{t+1}-V_t||_\infty < \varepsilon$

A simple example

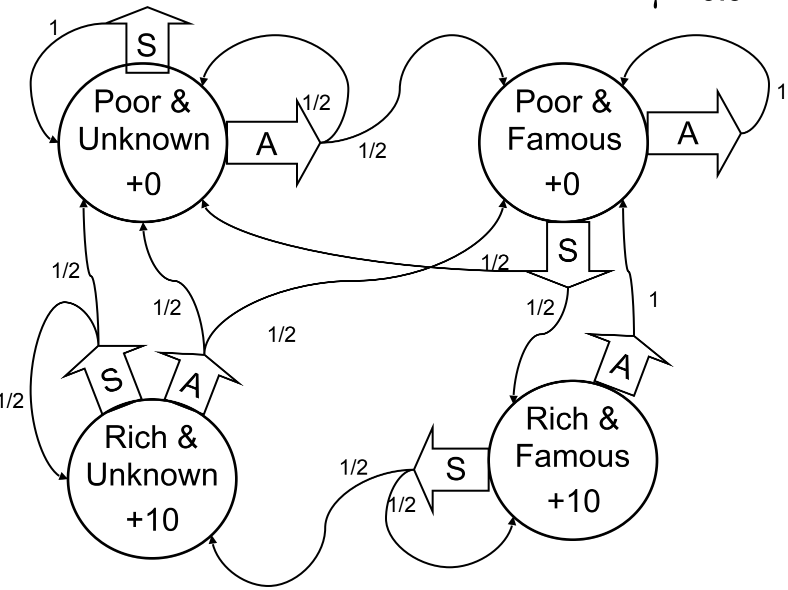
You run a startup company.

In every state you must choose between Saving money or Advertising.



Let's compute $V_t(x)$ for our example

$\gamma = 0.9$



t	V^t(PU)	V^t(PF)	V^t(RU)	V^t(RF)
1	0	0	10	10
2	0	4.5	14.5	19
3	2.03	9.46	17.44	25.08
4	5.17	13.61	20.17	29.13
5	8.45	16.91	22.88	32.19
6	11.41	19.62	25.43	34.78
∞	31.59	38.60	44.02	54.02

$$V^{t+1}(\mathbf{x}) = \max_{\mathbf{a}} R(\mathbf{x},\mathbf{a}) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}' \mid \mathbf{x},\mathbf{a}) V^t(\mathbf{x}')$$

What you need to know

- What's a Markov decision process
  - state, actions, transitions, rewards
  - a policy
  - value function for a policy
    - computing V_π
- Optimal value function and optimal policy
  - Bellman equation
- Solving Bellman equation
  - with value iteration, policy iteration and linear programming

In closing….

Recap

- Learning is function approximation
- Point estimation
- **Linear Least Squares Regression**
- **Regularization, Ridge, LASSO**
- **Model assessment, Bias-Variance tradeoff**
- **Cross validation, Bootstrap**
- (Non-)Convex optimization
- **Stochastic gradient descent, coordinate descent**
- Online learning (streaming), Perceptron
- **Logistic regression**
- **Support vector machine (SVM)**
- **Kernel trick**
- Intro to learning theory
- Supervised v. Unsupervised learning
- K-means
- Expectation-maximization (EM)
- Mixtures of Gaussians
- **Dimensionality reduction, PCA, matrix factorization**
- **Matrix completion**
- Neural networks, deep learning
- Recurrent neural networks for variable length sequences
- Text and document processing
- A/B testing, multi-armed bandits, active learning
- MDPs, Reinforcement learning

- HANDS ON EXPERIENCE…..

Please don't forget to fill out class evaluation on **myUW**.

If not registered, or want to give content related feedback, or you want just me to know something (anonymously), google form: https://tinyurl.com/y87hck25