Active Learning

CS329H: Machine Learning from Human Preferences

Stephan Sharkov, Rehaan Ahmad, Adarsh Jeewajee

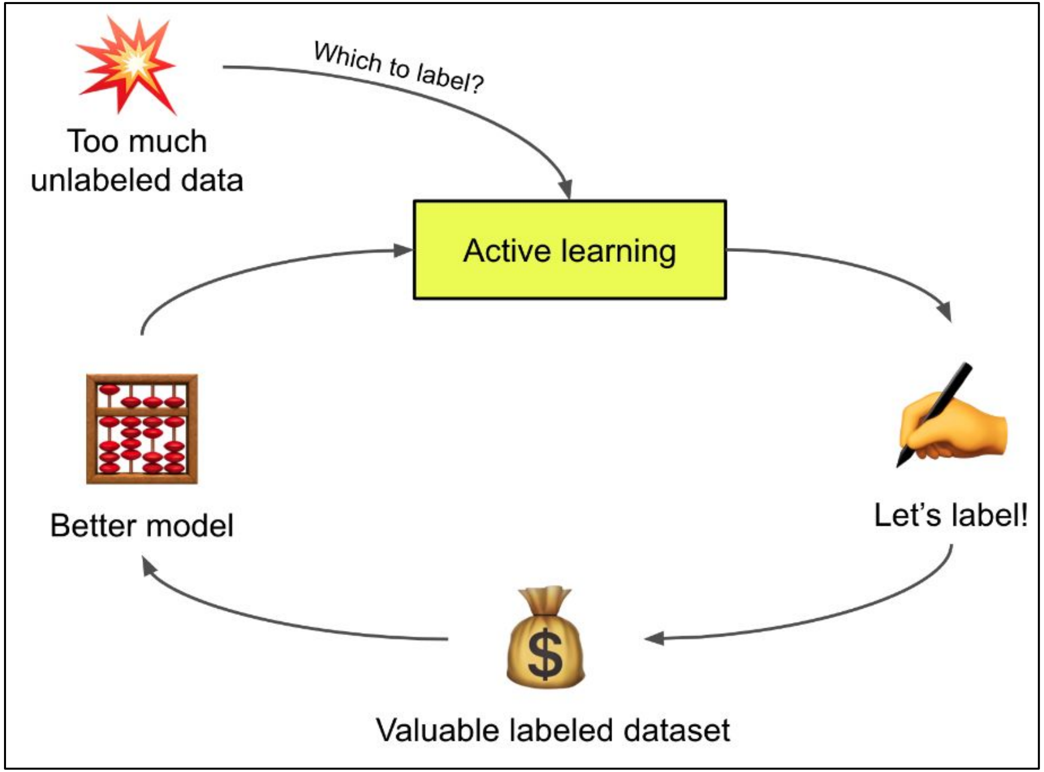
Plan

1. Active Learning Introduction
2. Active Learning and Pairwise Comparison
    a. Active Comparison-Based Learning in Ranking
    b. Active Learning Incorporating the User
3. Active Learning for Reward Functions
    a. Non-Batch methods
    b. Batch method

Plan

1. Active Learning Introduction
2. Active Learning and Pairwise Comparison
    a. Active Comparison-Based Learning in Ranking
    b. Active Learning Incorporating the User
3. Active Learning for Reward Functions
    a. Non-Batch methods
    b. Batch method

Active learning



Using:

- the current model
- *unlabelled* input distr P(x)

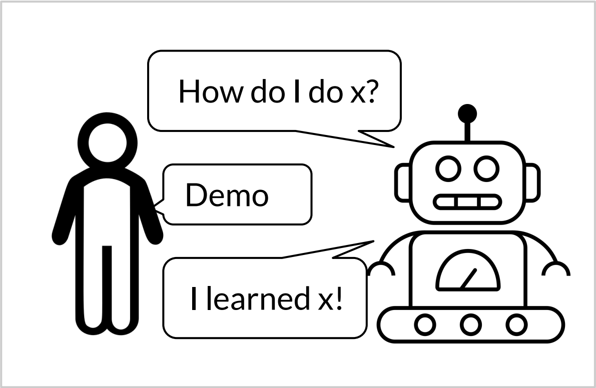
Determine which new point, if:

- labeled for a cost
- trained on

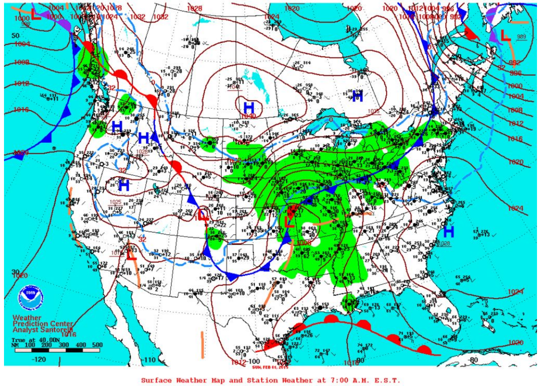
would maximize some model performance metric.

Useful if data collection difficult / costly.
Lower data requirements if queries chosen well.

Real-world scenarios



Learning unknown robot skills

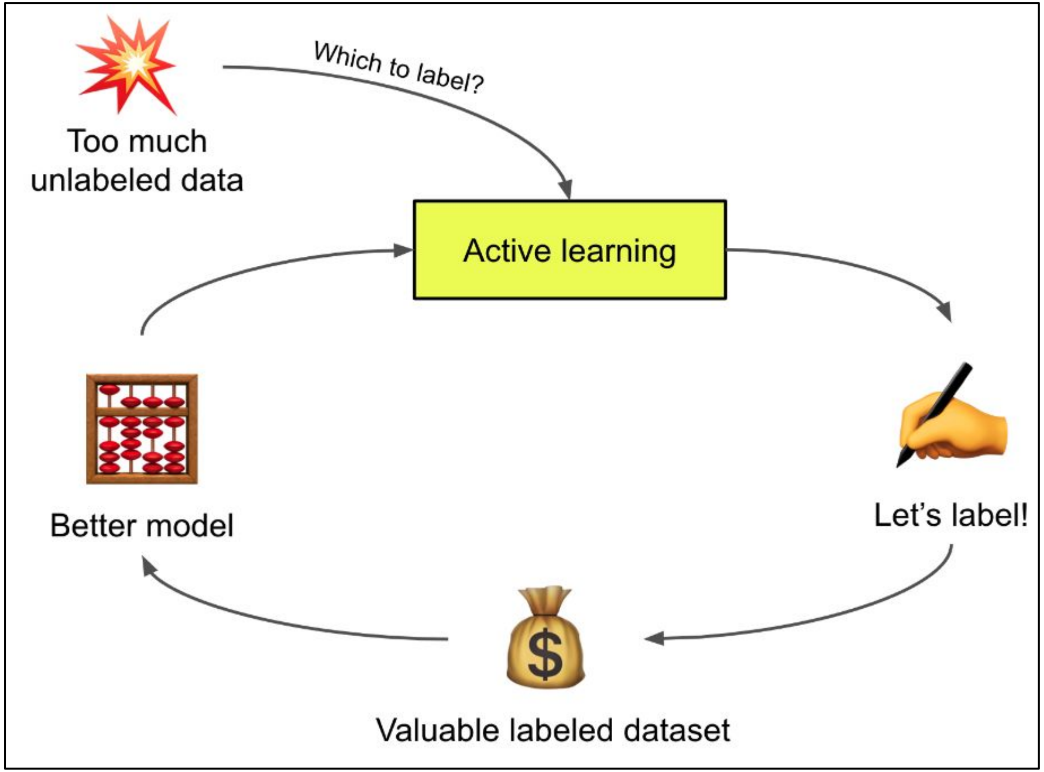


Sensor placement for enhanced predictions

LLM knowledge acquisition

Paper: Active Learning with Statistical Models [Cohn 96]

(Task: regression)



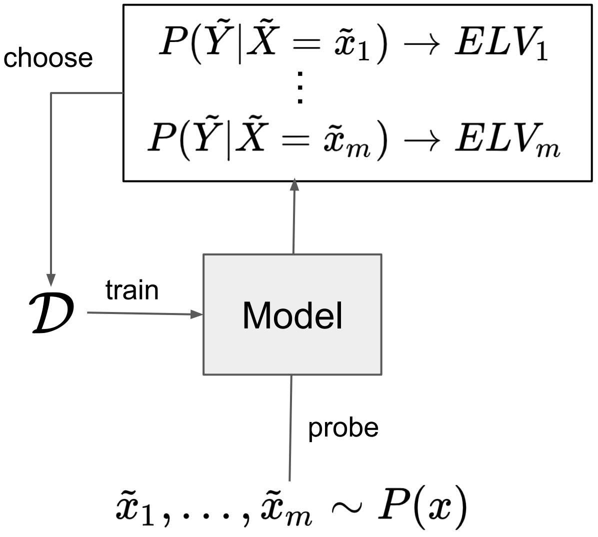
Expected model change
Expected error reduction
Variance reduction
Uncertainty reduction
Ensemble disagreement reduction
Diversity increase
Conformal prediction

Paper: Active Learning with Statistical Models [Cohn 96]

Which new point $\tilde{x} \sim P(x)$,
for which the model currently believes $P(\tilde{Y}|\tilde{X} = \tilde{x})$,
if annotated as $y(\tilde{x})$ and added to the training data $\mathcal{D}$,
would *potentially* lower the *expected variance of the learner*
across $P(x)$?

Without the true label $y(\tilde{x})$ and without re-training,
we visualize a hypothetical future where the model
is actually trained on $(\tilde{x}, f(\tilde{x}))$,
using $P(\tilde{Y}|\tilde{X} = \tilde{x})$ as a proxy for $y(\tilde{x})$.

Paper: Active Learning with Statistical Models [Cohn 96]



Which new point $\tilde{x} \sim P(x)$,
for which the model currently believes $P(\tilde{Y}|\tilde{X} = \tilde{x})$,
if annotated as $y(\tilde{x})$ and added to the training data $\mathcal{D}$,
would *potentially* lower the *expected variance of the learner*
across $P(x)$?

Without the true label $y(\tilde{x})$ and without re-training,
we visualize a hypothetical future where the model
is actually trained on $(\tilde{x}, f(\tilde{x}))$,
using $P(\tilde{Y}|\tilde{X} = \tilde{x})$ as a proxy for $y(\tilde{x})$.

Paper: Active Learning with Statistical Models [Cohn 96]

Next:

1. Mathematical description of expected learner variance (ELV)
2. Active learning algorithm to minimize ELV (for any model)
3. Closed-form ELV for 2 specific models
   (Gaussian mixture, locally-weighted regression)

Learner variance at x

Expected error at x (across choosable datasets and annotations)

$$\mathbb{E}[\ (\hat{y}(x;\mathcal{D}) - y(x))^2\ ]$$

$P(Y|X=x)$ → Stochasticity in annotations

$P(\mathcal{D})$ → D being picked, P(D) may be very different from P(x,y)

$$= \mathbb{E}_{y|x}[(y(x) - \mathbb{E}[y|x])^2]$$ → Noise in annotation of x (independent of learner)

$$+ (\mathbb{E}_{\mathcal{D}}[\hat{y}(x;\mathcal{D})] - \mathbb{E}[y|x])^2$$ → Learner squared bias at x

$$+ \mathbb{E}_{\mathcal{D}}[(\hat{y}(x;\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[\hat{y}(x;\mathcal{D})])^2]$$ → Learner variance at x

Expected learner variance

Learner variance at x

$$\sigma^2_{\hat{y}}(x, \mathcal{D}) = \mathbb{E}_{\mathcal{D}}[(\hat{y}(x;\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[\hat{y}(x;\mathcal{D})])^2]$$

$$\bar{\mathcal{D}} = \mathcal{D} \cup (\tilde{x}, \tilde{y})$$

Learner variance at x, if add $(\tilde{x}, \tilde{y})$ to training set

$$\tilde{\sigma}^2_{\hat{y}}(x, \bar{\mathcal{D}}) = \mathbb{E}_{\bar{\mathcal{D}}}[(\hat{y}(x;\bar{\mathcal{D}}) - \mathbb{E}_{\bar{\mathcal{D}}}[\hat{y}(x;\bar{\mathcal{D}})])^2]$$

$$\langle\tilde{\sigma}^2_{\hat{y}}(x, \mathcal{D})\rangle = \mathbb{E}_{P(\tilde{Y}|\tilde{X}=\tilde{x})}[\tilde{\sigma}^2_{\hat{y}}]$$

Expected learner variance at x across estimated label belief

$$\int_x P(x)\langle\tilde{\sigma}^2_{\hat{y}}(x, \mathcal{D})\rangle dx$$

Expected learner variance across input distr

ALgorithm to minimize ELV

$$\underset{\tilde{x} \in P(x)}{arg\,min} \int_x P(x) \langle \tilde{\sigma}^2_{\hat{y}}(x, \mathcal{D}) \rangle dx$$

$\mathcal{D}$= {} or init randomly

Each iter:

Sample candidate points $\tilde{x}_1, \ldots, \tilde{x}_m$ from $P(x)$

Compute current belief $P(\tilde{Y}|\tilde{X} = \tilde{x})$ for each point.

Calculate $\langle \tilde{\sigma}^2_{\hat{y}}(x, \mathcal{D}) \rangle$ on each point using belief

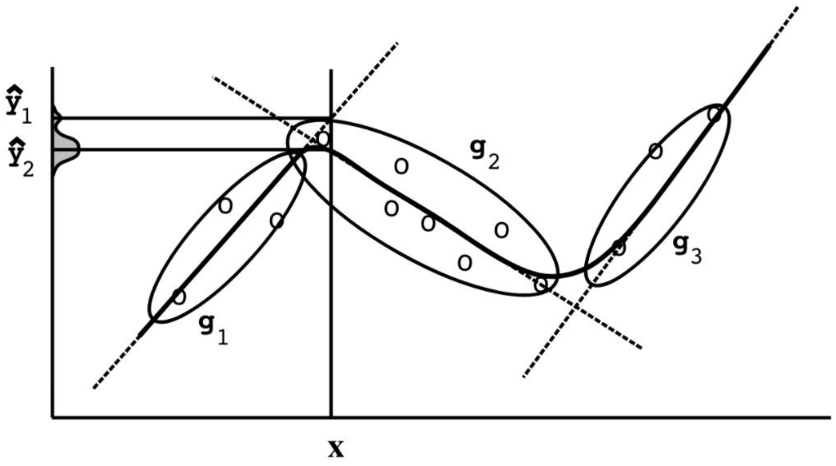
Calculate ELV integral via Monte Carlo sampling from $P(x)$

Label point with lowest ELV: $\tilde{x}^* \rightarrow y(\tilde{x}^*)$

$\mathcal{D} = \mathcal{D} \cup \{(\tilde{x}^*, y(\tilde{x}^*))\}$

Retrain on $\mathcal{D}$

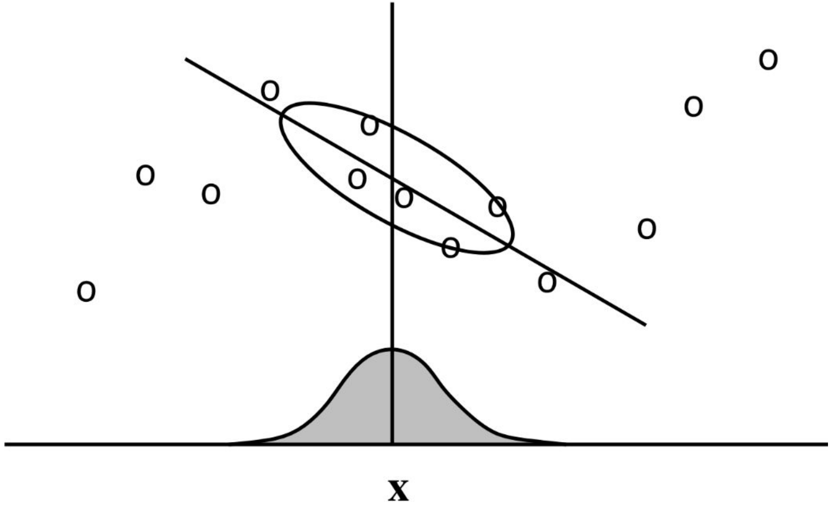
Models with closed form expected learner variance



$$P(x, y|i) = \frac{1}{2\pi\sqrt{|\Sigma_i|}} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)\right]$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mu_i = \begin{bmatrix} \mu_{x,i} \\ \mu_{y,i} \end{bmatrix} \quad \mathbf{\Sigma}_i = \begin{bmatrix} \sigma^2_{x,i} & \sigma_{xy,i} \\ \sigma_{xy,i} & \sigma^2_{y,i} \end{bmatrix}.$$

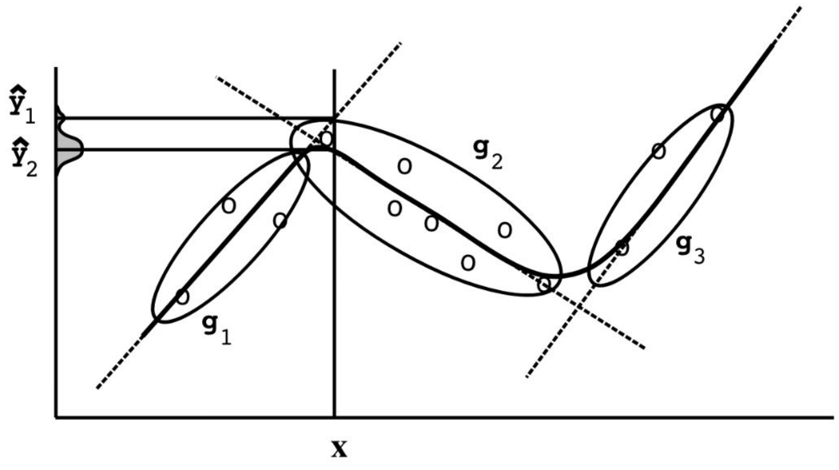
Mixture of gaussians



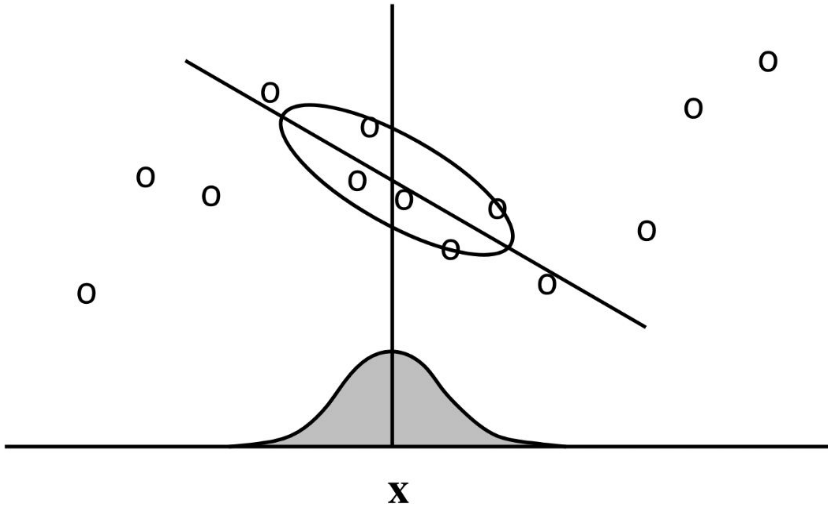
$$h_i(x) \equiv h(x - x_i) = \exp(-k(x - x_i)^2)$$

Locally-weighted regression

Models with closed form expected learner variance



$$\left\langle \tilde{\sigma}^2_{\hat{y}} \right\rangle = \sum_{i=1}^{N} \frac{h_i^2 \left\langle \tilde{\sigma}^2_{y|x,i} \right\rangle}{n_i + \tilde{h}_i} \left(1 + \frac{(x - \mu_{x,i})^2}{\sigma^2_{x,i}} \right)$$



$$\left\langle \tilde{\sigma}^2_{\hat{y}} \right\rangle = \frac{\left\langle \tilde{\sigma}^2_{y|x} \right\rangle}{(n + \tilde{h})^2} \left[\sum_i h_i^2 + \tilde{h}^2 + \frac{(x - \tilde{\mu}_x)^2}{\tilde{\sigma}^2_x} \left(\sum_i h_i^2 \frac{(x_i - \tilde{\mu}_x)^2}{\tilde{\sigma}^2_x} + \tilde{h}^2 \frac{(\tilde{x} - \tilde{\mu}_x)^2}{\tilde{\sigma}^2_x} \right) \right]$$

Experiment contrasting both models / algorithms

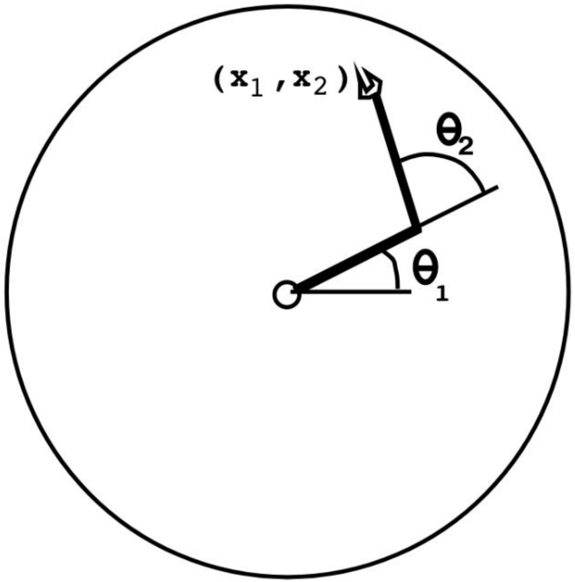


Figure 4: The arm kinematics problem. The learner attempts to predict tip position given a set of joint angles (θ_1, θ_2).

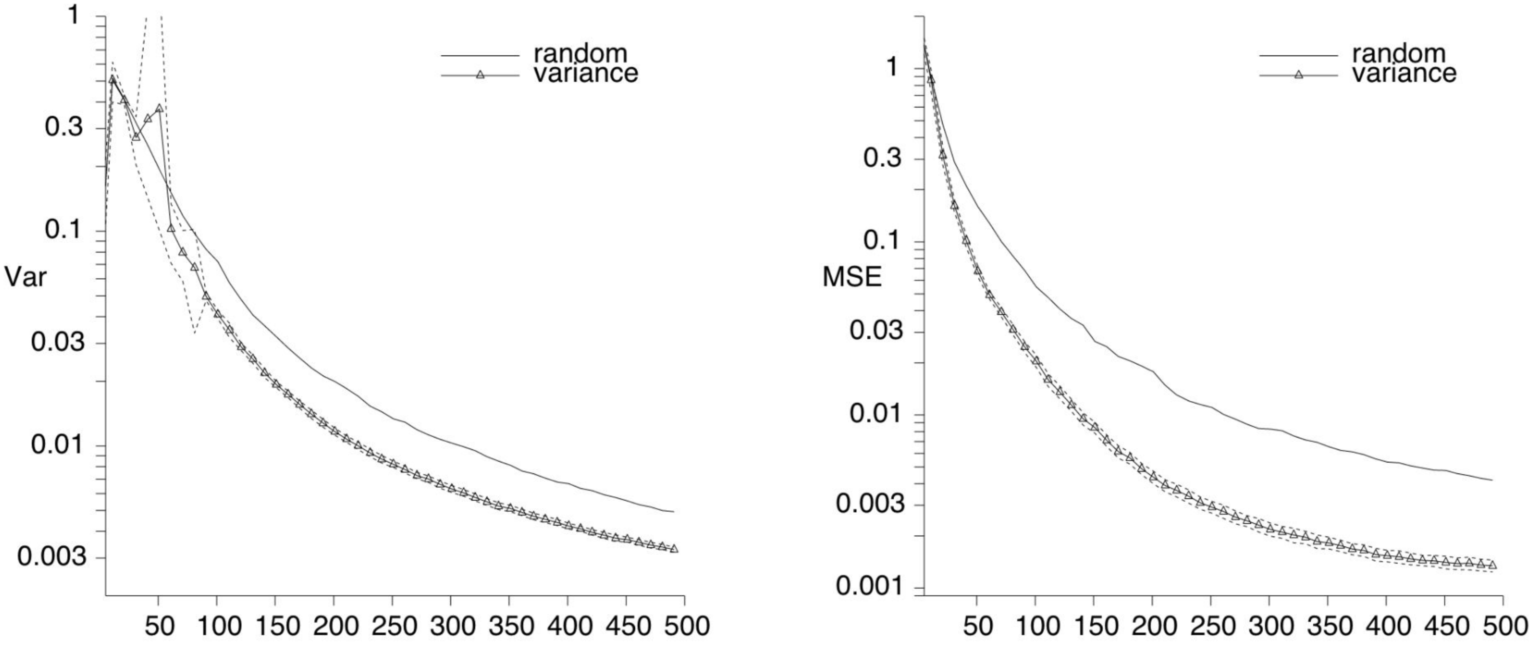


Figure 5: Variance and MSE learning curves for mixture of 60 Gaussians trained on the Arm2D domain. Dotted lines denote standard error for average of 10 runs, each started with one initial random example.

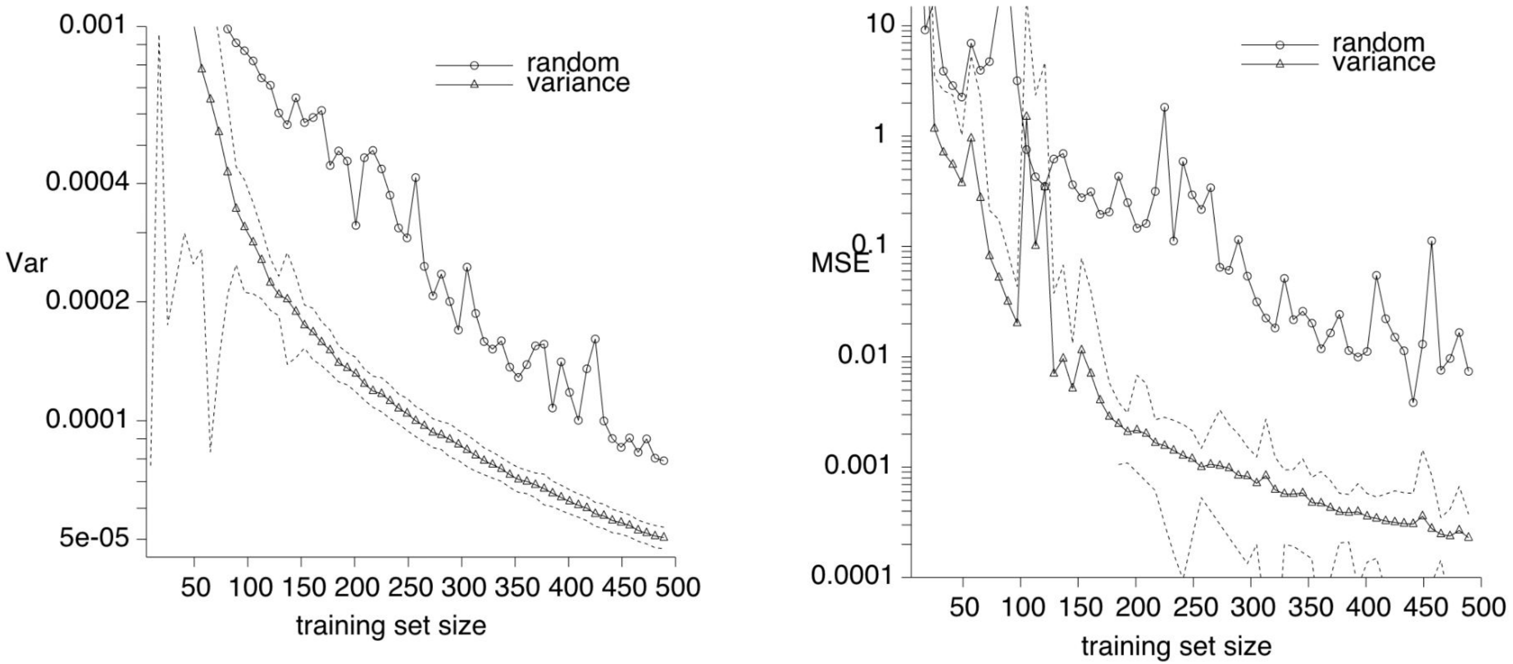


Figure 6: Variance and MSE learning curves for LOESS model trained on the Arm2D domain. Dotted lines denote standard error for average of 60 runs, each started with a single initial random example.

Plan

1. Active Learning Introduction
2. Active Learning and Pairwise Comparison
    a. Active Comparison-Based Learning in Ranking
    b. Active Learning Incorporating the User
3. Active Learning for Reward Functions
    a. Non-Batch methods
    b. Batch method

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Problem Definition

A ranking over a set of **n** objects $\Theta = (\theta_1, \theta_2, \ldots, \theta_n)$ is a mapping $\sigma : \{1,\ldots,n\} \rightarrow \{1,\ldots,n\}$ that prescribes an order

$\sigma(\Theta) := \theta_{\sigma(1)} \prec \theta_{\sigma(2)} \prec \cdots \prec \theta_{\sigma(n-1)} \prec \theta_{\sigma(n)}$ where $\theta_i \prec \theta_j$ means θ_i precedes θ_j in the ranking.

- Total possible number of rankings is **n!**
- Ranking of **n** objects can be done with standard sorting methods using **nlogn** pairwise comparisons, if the comparisons are picked at random for every query $q_{i,j} := \{\theta_i \prec \theta_j\}$
- Find a way to decrease the **nlogn**, this specific case **dlogn**

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Reasoning and Motivation

- Most work in ranking assumes a passive approach of doing all the rankings
- This might be inefficient since some comparisons don’t give that much value, and humans can be costly
- Applications of this are quite popular and common
- What if we take the idea of human knowledge and transfer it to machine
- Specifically consider the geometric approach of using the embedding space

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Notations and assumptions

- Objects can be embedded in $\mathbf{R}^d$
- $\theta_1, \ldots, \theta_n$ their locations in $\mathbf{R}^d$
- Every ranking σ can be specified by a reference point $r_\sigma \in \mathbf{R}^d$, if the σ ranks $\theta_i \prec \theta_j$, then $\|\theta_i - r_\sigma\| < \|\theta_j - r_\sigma\|$
- $\Sigma_{n,d}$ - set of all possible rankings of the **n** objects that satisfy this embedding condition
- $M_n(\sigma)$ - the number of pairwise comparisons to identify the ranking σ. We will reason about $E[M_n]$
- $q_{i,j}$ - the query of comparison between objects **i** and **j**

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Embedding Space

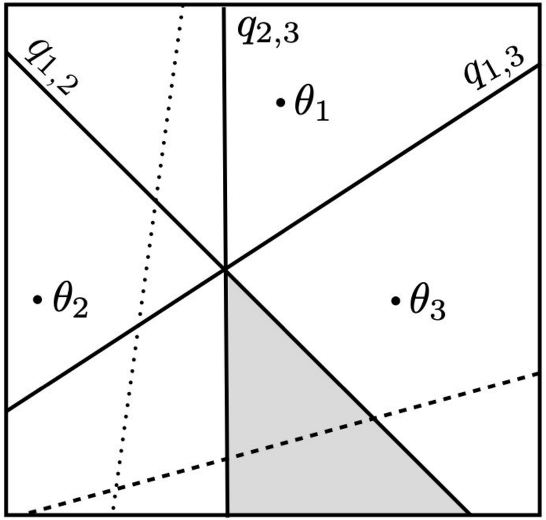


Figure 2: Objects $\theta_1, \theta_2, \theta_3$ and queries. The r_σ lies in the shaded region (consistent with the labels of $q_{1,2}, q_{1,3}, q_{2,3}$). The dotted (dashed) lines represent new queries whose labels are (are not) ambiguous given those labels.

- label of $\mathbf{q_{i,j}}$ is binary and denoted as $\mathbf{y_{i,j} := 1\{q_{i,j}\}}$ **(e.g. $\mathbf{y_{1,2}=0, y_{3,2}=1}$)**

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Embedding Space

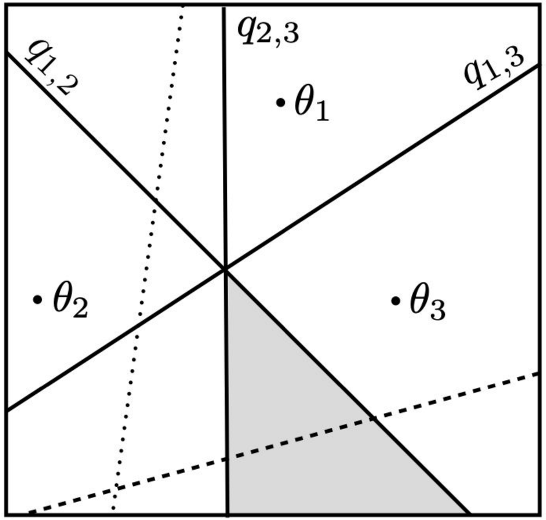


Figure 2: Objects $\theta_1, \theta_2, \theta_3$ and queries. The r_σ lies in the shaded region (consistent with the labels of $q_{1,2}, q_{1,3}, q_{2,3}$). The dotted (dashed) lines represent new queries whose labels are (are not) ambiguous given those labels.

- label of $\mathbf{q_{i,j}}$ is binary and denoted as $\mathbf{y_{i,j} := 1\{q_{i,j}\}}$ **(e.g. $\mathbf{y_{1,2}=0, y_{3,2}=1}$)**
- The dotted/dashed lines represents potential queries of one of the three elements with a new element

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Embedding Space

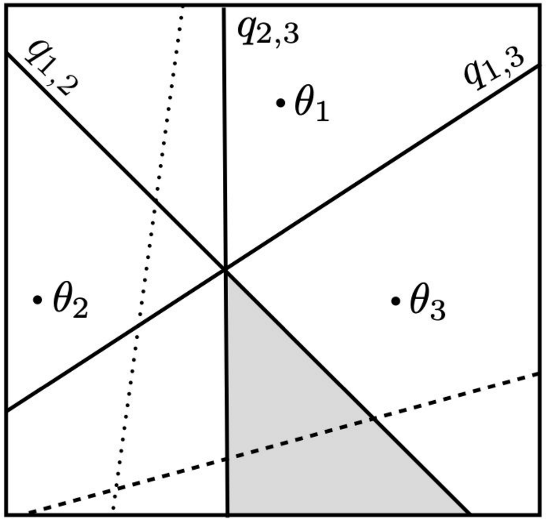


Figure 2: Objects $\theta_1, \theta_2, \theta_3$ and queries. The r_σ lies in the shaded region (consistent with the labels of $q_{1,2}, q_{1,3}, q_{2,3}$). The dotted (dashed) lines represent new queries whose labels are (are not) ambiguous given those labels.

- label of $\mathbf{q_{i,j}}$ is binary and denoted as $\mathbf{y_{i,j} := 1\{q_{i,j}\}}$ **(e.g. $y_{1,2}$=0, $y_{3,2}$=1)**
- The dotted/dashed lines represents potential queries of one of the three elements with a new element
- With the **dotted line label of the query might be different** depending on whether we are comparing with 2 (label 0) or 1 and 3 (label 1). Thus, we need an actual comparison to know for sure where that element stands in the ranking.
- With the **dashed line label of the query is always 1** and it can be inferred using the labels of other queries

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Embedding Space

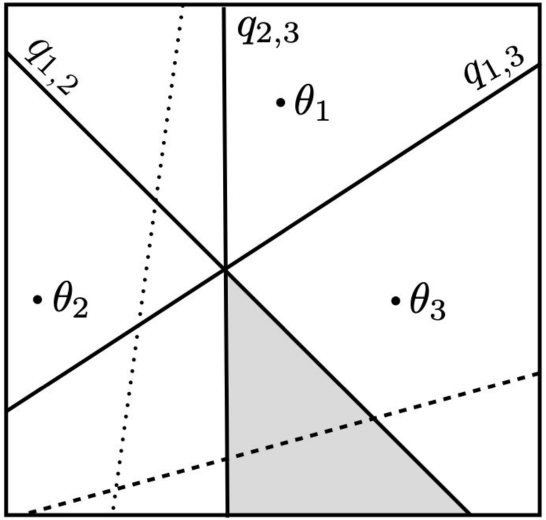


Figure 2: Objects $\theta_1, \theta_2, \theta_3$ and queries. The r_σ lies in the shaded region (consistent with the labels of $q_{1,2}, q_{1,3}, q_{2,3}$). The dotted (dashed) lines represent new queries whose labels are (are not) ambiguous given those labels.

- label of $\mathbf{q_{i,j}}$ is binary and denoted as $\mathbf{y_{i,j} := 1\{q_{i,j}\}}$ **(e.g. $q_{1,2}$=0, $q_{2,3}$=1)**
- The dotted/dashed lines represents potential queries of one of the three elements with a new element
- With the **dotted line label of the query might be different** depending on whether we are comparing with 2 (label 0) or 1 and 3 (label 1). Thus, we need an actual comparison to know for sure where that element stands in the ranking.
- With the **dashed line label of the query is always 1** and it can be inferred using the labels of other queries
- **Dotted lines is the queries for which we actually want to use the human**

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Sequential Algorithm

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$

initialize: objects $\theta_1, \ldots, \theta_n$ in uniformly random order

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 1: Sequential algorithm for selecting queries. See Figure 2 and Section 4.2 for the definition of an ambiguous query.

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Sequential Algorithm

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$
initialize: objects $\theta_1, \ldots, \theta_n$ in uniformly random order

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 1: Sequential algorithm for selecting queries. See Figure 2 and Section 4.2 for the definition of an ambiguous query.

Definition 3. *[9] Let S be a finite subset of $\mathbb{R}^d$ and let $S^+ \subset S$ be points labeled $+1$ and $S^- = S \setminus S^+$ be the points labeled -1 and let x be any other point except the origin. If there exists two homogeneous linear separators of S^+ and S^- that assign different labels to the point x, then the label of x is said to be* ambiguous with respect to *S*.

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Sequential Algorithm

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$
initialize: objects $\theta_1, \ldots, \theta_n$ in uniformly random order

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 1: Sequential algorithm for selecting queries. See Figure 2 and Section 4.2 for the definition of an ambiguous query.

Definition 3. *[9] Let S be a finite subset of $\mathbb{R}^d$ and let $S^+ \subset S$ be points labeled $+1$ and $S^- = S \setminus S^+$ be the points labeled -1 and let x be any other point except the origin. If there exists two homogeneous linear separators of S^+ and S^- that assign different labels to the point x, then the label of x is said to be* ambiguous with respect to *S*.

Σn,d - set of all possible rankings that satisfy the embedding conditions

Q(i,j) - number of rankings that exist for **i** elements in **j**-dimensional space (e.g. **Q(1,d) = 1** and **Q(n, 0) = 1**)

|Σn,d| = Q(n,d) = Q(n−1,d) + (n−1) * Q(n−1,d−1)
- follows a 1D idea of inserting a new element into a list of elements

|Σn,d| = Q(n,d) = Θ(n^{2d})

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Sequential Algorithm

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$
initialize: objects $\theta_1, \ldots, \theta_n$ in uniformly random order

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 1: Sequential algorithm for selecting queries. See Figure 2 and Section 4.2 for the definition of an ambiguous query.

Definition 3. *[9] Let S be a finite subset of $\mathbb{R}^d$ and let $S^+ \subset S$ be points labeled $+1$ and $S^- = S \setminus S^+$ be the points labeled -1 and let x be any other point except the origin. If there exists two homogeneous linear separators of S^+ and S^- that assign different labels to the point x, then the label of x is said to be* ambiguous with respect to *S*.

|Σn,d| - set of all possible rankings that satisfy the embedding conditions

Q(i,j) - number of rankings that exist for **i** elements in **j**-dimensional space (e.g. **Q(1,d) = 1** and **Q(n, 0) = 1**)

|Σn,d| = Q(n,d) = Q(n−1,d) + (n−1) * Q(n−1,d−1)
- follows a 1D idea of inserting a new element into a list of elements

|Σn,d| = Q(n,d) = Θ(n2d)

$$M_n = \sum_{k=1}^{n-1} \sum_{i=1}^{k} \mathbf{1}\{\text{Request } q_{i,k+1}\}$$

P(k,d) - number of rankings that are possible to be true for query with a new element **k+1**

Request$_{\text{qi, k+1}}$ = P(k,d)/Q(k,d)

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Sequential Algorithm

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$
initialize: objects $\theta_1, \ldots, \theta_n$ in uniformly random order

for j=2,. . . ,n
  for i=1,. . . ,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 1: Sequential algorithm for selecting queries. See Figure 2 and Section 4.2 for the definition of an ambiguous query.

Definition 3. *[9] Let S be a finite subset of $\mathbb{R}^d$ and let $S^+ \subset S$ be points labeled $+1$ and $S^- = S \setminus S^+$ be the points labeled -1 and let x be any other point except the origin. If there exists two homogeneous linear separators of S^+ and S^- that assign different labels to the point x, then the label of x is said to be* ambiguous with respect to S.

|Σn,d| - set of all possible rankings that satisfy the embedding conditions

Q(i,j) - number of rankings that exist for **i** elements in **j**-dimensional space (e.g. **Q(1,d) = 1** and **Q(n, 0) = 1**)

|Σn,d| = Q(n,d) = Q(n−1,d) + (n−1) * Q(n−1,d−1) - follows a 1D idea of inserting a new element into a list of elements

|Σn,d| = Q(n,d) = Θ(n2d)

$$M_n = \sum_{k=1}^{n-1} \sum_{i=1}^{k} \mathbf{1}\{\text{Request } q_{i,k+1}\}$$

P(k,d) - number of rankings that are possible to be true for query with a new element **k+1**

$\text{Request}_{qi,\ k+1}$ = P(k,d)/Q(k,d)

High value means the probability of this request being ambiguous is high, so higher probability of it getting picked

$E[M_n]$ = O(2dlogn)

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Sequential Algorithm

Query Selection Algorithm

input: n objects in $\mathbb{R}^d$
initialize: objects $\theta_1, \ldots, \theta_n$ in uniformly random order

for j=2,...,n
  for i=1,...,j-1
    if $q_{i,j}$ is *ambiguous*,
      request $q_{i,j}$'s label from reference;
    else
      impute $q_{i,j}$'s label from previously labeled queries.

output: ranking of n objects

Figure 1: Sequential algorithm for selecting queries. See Figure 2 and Section 4.2 for the definition of an ambiguous query.

Definition 3. *[9] Let S be a finite subset of $\mathbb{R}^d$ and let $S^+ \subset S$ be points labeled $+1$ and $S^- = S \setminus S^+$ be the points labeled -1 and let x be any other point except the origin. If there exists two homogeneous linear separators of S^+ and S^- that assign different labels to the point x, then the label of x is said to be* ambiguous with respect to S.

|Σn,d| - set of all possible rankings that satisfy the embedding conditions

Q(i,j) - number of rankings that exist for **i** elements in **j**-dimensional space (e.g. **Q(1,d) = 1** and **Q(n, 0) = 1**)

|Σn,d| = Q(n,d) = Q(n−1,d) + (n−1) * Q(n−1,d−1) - follows a 1D idea of inserting a new element into a list of elements

|Σn,d| = Q(n,d) = Θ(n2d)

$M_n = \sum_{k=1}^{n-1} \sum_{i=1}^{k} \mathbf{1}\{\text{Request } q_{i,k+1}\}$

P(k,d) - number of rankings that are possible to be true for query with a new element **k+1**

$\text{Request}_{qi,\,k+1}$ = P(k,d)/Q(k,d)

High value means the probability of this request being ambiguous is high, so higher probability of it getting picked

$E[M_n]$ = O(2dlogn)

But what if that one human is not good?

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Robust Sequential Algorithm

- Same idea but uses majority voting
- However, a group of people can still consistently give incorrect response
- Thus, the authors are hoping that majority voting can get at least a partial ranking of the objects

Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Results

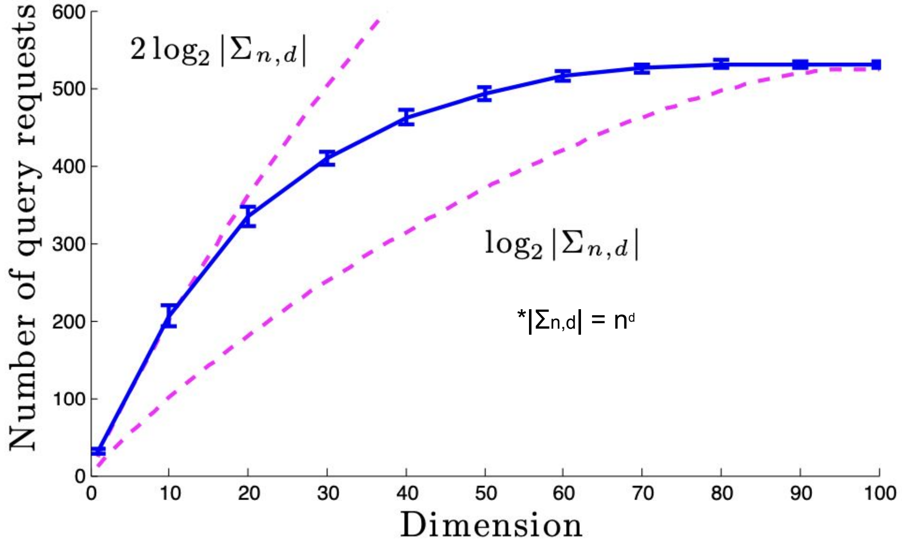


Figure 3: Mean and standard deviation of requested queries (solid) in the noiseless case for $n = 100$; $\log_2|\Sigma_{n,d}|$ is a lower bound (dashed).

Table 1: Statistics for the algorithm robust to persistent noise of Section 5 with respect to all $\binom{n}{2}$ pairwise comparisons. Recall y is the noisy response vector, $\tilde{y}$ is the embedding's solution, and $\hat{y}$ is the output of the robust algorithm.

Dimension		2	3
% of queries requested	mean	14.5	18.5
	std	5.3	6
Average error	$d(y, \tilde{y})$	0.23	0.21
	$d(y, \hat{y})$	0.31	0.29

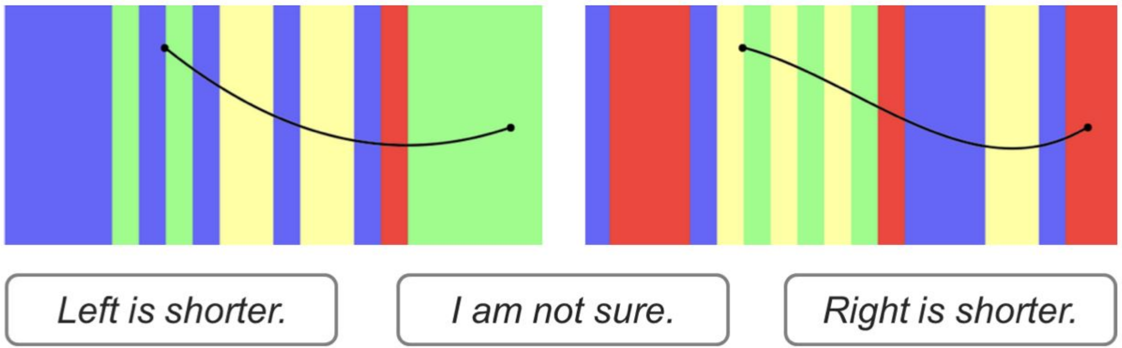
Jamieson, Kevin G., and Robert Nowak. "Active ranking using pairwise comparisons." *Advances in neural information processing systems* 24 (2011).

Plan

1. Active Learning Introduction
2. Active Learning and Pairwise Comparison
    a. Active Comparison-Based Learning in Ranking
    b. Active Learning Incorporating the User
3. Active Learning for Reward Functions
    a. Non-Batch methods
    b. Batch method

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

What if Active Learning knowledge did not come from data or embeddings but directly from the user?



Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

Problem Formulation

c - cost function to be determined with fewest possible questions

H - given discrete set of hypothesis

$\mathbf{C_h}$ - cost function associated with hypothesis **h**

h* - true hypothesis

t(x,y) - performed test on comparing **x** and **y**

O = x > y or x < y - observation from **t(x,y)**

$\mathbf{S = \{(t_1, o_1), \ldots (t_m, o_m)\}}$ - sequence of **m** tests and observations

w(H | S) - probability mass of all hypothesis that are still consistent

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

Problem Formulation

c - cost function to be determined with fewest possible questions

H - given discrete set of hypothesis

C_h - cost function associated with hypothesis **h**

h* - true hypothesis

t(x,y) - performed test on comparing **x** and **y**

O = x > y or x < y - observation from **t(x,y)**

S = {(t_1, o_1), … (t_m, o_m)} - sequence of **m** tests and observations

w(H, S) - probability mass of all hypothesis that are still consistent

Noiseless setting:

$$w(\mathcal{H} \mid \mathcal{S}) = \sum_{h \in \mathcal{H}} w(h \mid \mathcal{S})$$

$$w(h \mid \mathcal{S}) = p(\mathcal{S} \mid h)$$

We assume that the sequence of test-observation pairs (t, o) in $\mathcal{S}$ is independent:

$$p(\mathcal{S} \mid h) = \prod_{(t,o) \in \mathcal{S}} p((t, o) \mid h)$$

In the noiseless setting, we assume the user always selects the item that minimizes their cost function:

$$p((t, o = x) \mid h) = \begin{cases} 1 & c_h(x) < c_h(y) \\ 0 & \text{else} \end{cases}$$

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

User Noise Modeling

- Users are not perfect, which may result in poor performance
- Majority of prior research applied noise to all queries to account for user
- This is not an accurate representation of real-world behaviour, noise should be “query-dependent” - supported in psychology literature
- Prior literature derived logistic model based on Luce-Sheppard’s rule to account for the noise
- $p((t, o = x) \mid h) \propto \exp(-\gamma * c_h(x))$

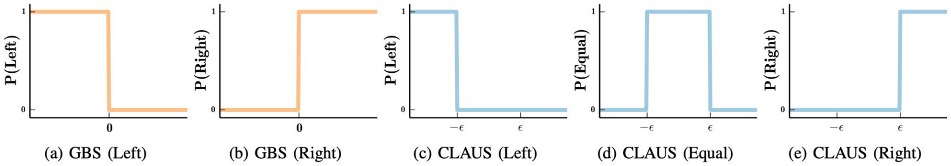


Fig. 2: User response model in the noiseless setting

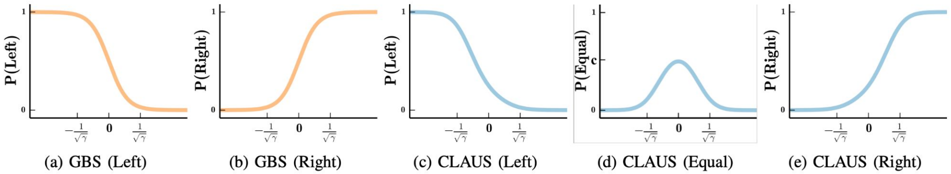


Fig. 3: Luce Sheppard noise model

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

CLAUS

- Allowing users to express uncertainty will increase satisfaction and algorithm efficiency
- Using cost function **c**, uncertainty is **ε** where if **|c(x) − c(y)| < ε** the user is uncertain between **x** and **y**
- More insight into user's cost function, as we know that **x** and **y** are similar
- Observation space: **O(t) = (x, y, $\widetilde{xy}$)**, where **$\widetilde{xy}$** - uncertain response
- Learn **(c, ε)** pair, but objective is on **c** only

Let c_h be the cost function of the hypothesis, and ϵ_h is the uncertainty parameter. In the noiseless setting, we assume user response corresponds to:

$$p((t, o = x) \mid h) = \begin{cases} 1 & c_h(x) < c_h(y) - \epsilon_h \\ 0 & \text{else} \end{cases}$$

$$p((t, o = \widetilde{xy}) \mid h) = \begin{cases} 1 & |c_h(x) - c_h(y)|^2 < \epsilon_h^2 \\ 0 & \text{else} \end{cases}$$

We extend this to model noise, stemming from the Luce-Sheppard model from Sec. II-C.

$$p((t, o = x) \mid h) \propto \exp\left(-\gamma(c_h(x) - c_h(y))\right)$$

$$p((t, o = \widetilde{xy}) \mid h) \propto \exp\left(-\frac{1}{\epsilon_h^2}\left[c_h(x) - c_h(y)\right]^2\right) c$$

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

Equivalence Class Determination

- Rather than trying to find a specific pair of **(c, ε)** focus on finding an equivalent class (similar/indifferent hypothesis within the same space)
- Considered finding an equivalence class of different sizes
- Tests and information learned about **(c, ε)** are main factors of similarity search

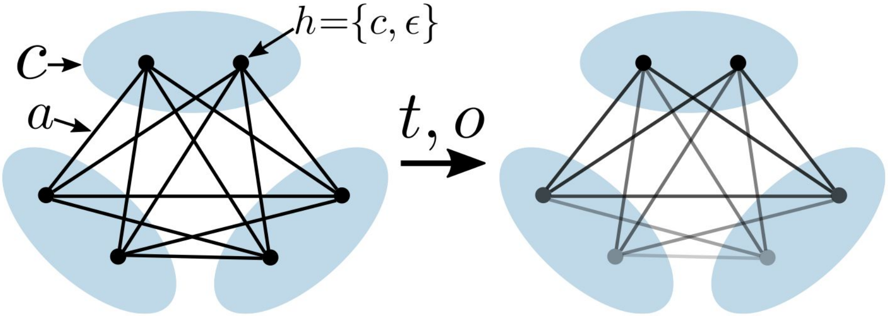


Fig. 4: CLAUS using EC^2. Each cost function c corresponds to an equivalence class (blue ellipse). Hypotheses (black dots) are $\{c, \epsilon\}$ pairs. Hypotheses sharing a cost c are said to be inside the equivalence class of c. The algorithm constructs a graph, drawing an edge (black line) between hypotheses in different equivalence classes. After performing a test and receiving an observation, the evidence results in downweighting some hypotheses, which in turn downweights the edges they connect to.

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

Experiments and Results

- Less queries, especially if more epsilons (bigger equivalence class size)
- Users enjoyed the CLAUS model more than GBS, but preference was almost the same
- Authors tested two versions of CLAUS but with mix of GBS
- Slightly lower accuracy

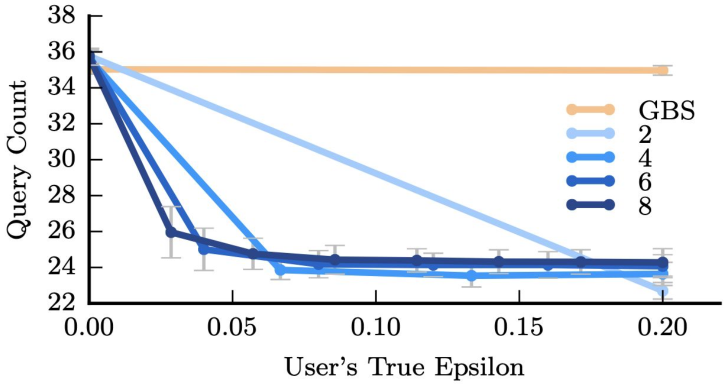


Fig. 5: We compare the affect of the number of epsilons on CLAUS's query count across the user's ϵ^*. We also show the query count of GBS. Note that when the user expresses any uncertainty, $\epsilon^* > 0$, all CLAUS methods utilize far fewer queries than GBS.

TABLE I: Accuracy and Query Count

Category	Accuracy	Query Count
GBS - About Equal	94.15 ± 0.52	36.02 ± 0.03
GBS - Not Sure	$\mathbf{94.66 \pm 0.55}$	35.95 ± 0.04
CLAUS - About Equal	91.56 ± 0.84	$\mathbf{25.93 \pm 0.41}$
CLAUS - Not Sure	90.86 ± 0.74	26.98 ± 0.47

TABLE II: CLAUS Parameters

Category	Marked Uncertainty	Epsilon
About Equal	7.80 ± 0.70	0.07 ± 0.00
Not Sure	5.57 ± 0.71	0.06 ± 0.01

Rachel Holladay and Shervin Javdani and Anca Dragan and Siddhartha Srinivasa. "Active Comparison Based Learning Incorporating User Uncertainty and Noise." Proceedings of RSS '16 Workshop on Model Learning for Human-Robot Communication (2016).

Plan

1. Active Learning Introduction
2. Active Learning and Pairwise Comparison
    a. Active Comparison-Based Learning in Ranking
    b. Active Learning Incorporating the User
3. Active Learning for Reward Functions
    a. Non-Batch methods
    b. Batch method

Active Preference-Based Learning of Reward Functions

Goal

Goal of the paper is to learn reward functions from human preferences for a dynamical system.

Typically in these settings a common approach to enforcing human preference is demonstrations with Inverse RL

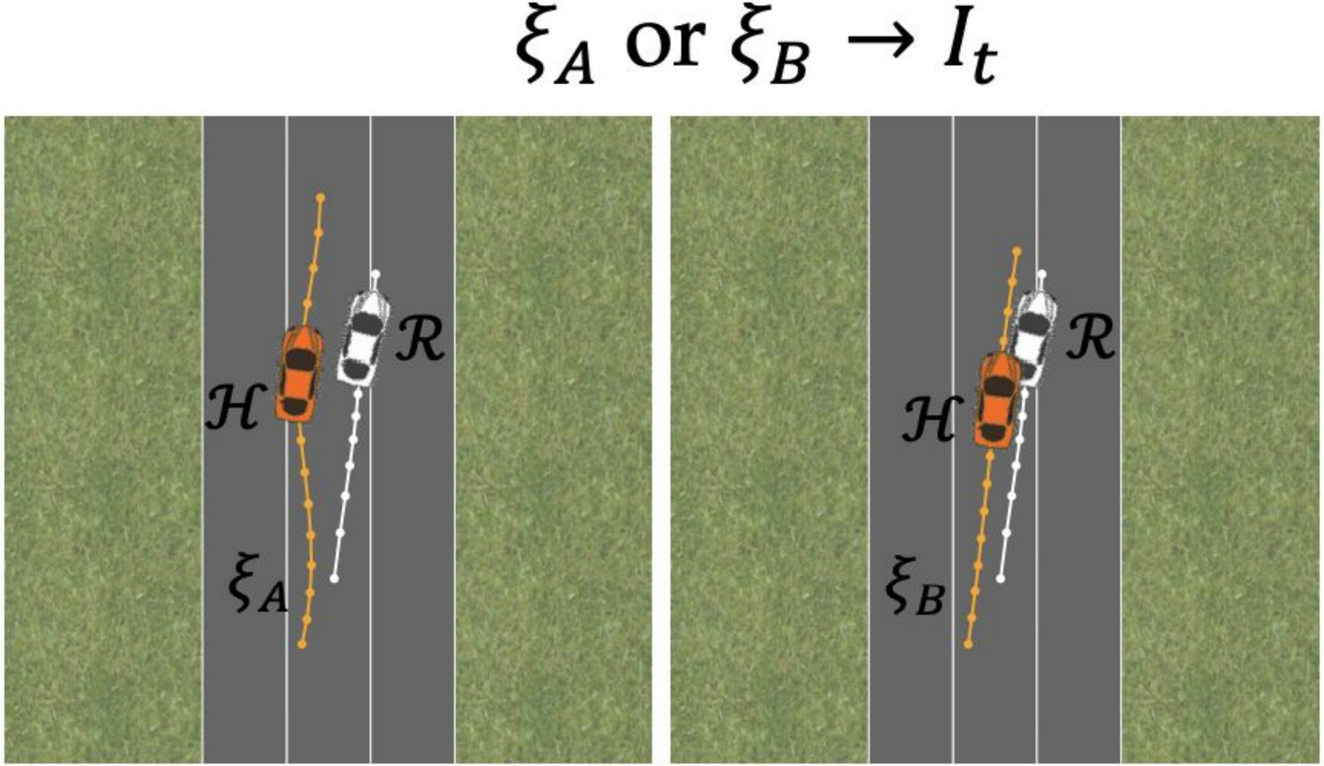
- VICE
- GAIL
- etc

What is a “dynamical system”?

$$x^{t+1} = f_{HR}(x^t, u_R, u_H). \quad (1)$$

We define a trajectory $\xi \in \Xi$, where $\xi = (x^0, u_R^0, u_H^0), \ldots, (x^N, u_R^N, u_H^N)$ is a finite horizon sequence

$$\phi(x^t, u_R^t, u_H^t) \in \mathbb{R}^d$$



(a) Preference query.

Dealing with Rewards

Assumption: Preference reward function can be represented as a linear combination of the features

- rH to represent desired human preference for that state

$$r_H(x^t, u_R^t, u_H^t) = \mathbf{w}^\top \phi(x^t, u_R^t, u_H^t),$$

Sum the rewards over an entire time series :

$$R_H(x^0, \mathbf{u}_R, \mathbf{u}_H) = \sum_{t=0}^{N} r_H(x^t, u^t, u_H^t)$$

$$\Phi = \sum_{t=0}^{N} \phi(x^t, u_R^t, u_H^t)$$

- So the reward over a trajectory would just be:

$$R_H(\xi) = \mathbf{w} \cdot \Phi(\xi)$$

How does p(w) look like?

The scale of **w** does not change the actual relative rewards produced with **w**:

- Constrain ||w|| <= 1
- w lies within a unit ball
- Initial prior is uniform over the unit ball

Incorporating softmax

$$p(I_t|\mathbf{w}) = \begin{cases} \frac{\exp(R_H(\xi_A))}{\exp(R_H(\xi_A))+\exp(R_H(\xi_B))} & I_t = +1 \\ \frac{\exp(R_H(\xi_B))}{\exp(R_H(\xi_A))+\exp(R_H(\xi_B))} & I_t = -1 \end{cases}$$

- Idea: Use $p(I_t|w)$ to do Bayesian update on $p(w)$
  - More on this later

$$p(\mathbf{w}|I_t) \propto p(\mathbf{w}) \cdot p(I_t|\mathbf{w}).$$

$$\varphi = \Phi(\xi_A) - \Phi(\xi_B)$$

$$f_\varphi(\mathbf{w}) = p(I_t|\mathbf{w}) = \frac{1}{1 + \exp(-I_t \mathbf{w}^\top \varphi)}$$

How do you generate queries?

Synthetically….

- “we want to find the next query such that it will help us remove as much volume (the integral of the unnormalized pdf over w) as possible from the space of possible rewards”

$$\max_{\varphi} \quad \min\{\mathbb{E}[1 - f_{\varphi}(\mathbf{w})], \mathbb{E}[1 - f_{-\varphi}(\mathbf{w})]\}$$
$$\text{subject to} \quad \varphi \in \mathbb{F}$$

$$f_{\varphi}(\mathbf{w}) = p(I_t|\mathbf{w}) = \frac{1}{1 + \exp(-I_t \mathbf{w}^{\top} \varphi)}$$

$$\mathbb{F} = \{\varphi : \varphi = \Phi(\xi_A) - \Phi(\xi_B), \xi_A, \xi_B \in \Xi, \tau = (x^0, \mathbf{u}_R^A) = (x^0, \mathbf{u}_R^B)\}$$

$$\max_{x^0, \mathbf{u}_R, \mathbf{u}_H^A, \mathbf{u}_H^B} \quad \min\{\mathbb{E}[1 - f_{\varphi}(\mathbf{w})], \mathbb{E}[1 - f_{-\varphi}(\mathbf{w})]\}$$

How to optimize?

$$\max_{x^0, \mathbf{u}_R, \mathbf{u}_H^A, \mathbf{u}_H^B} \quad \min\{\mathbb{E}[1 - f_\varphi(\mathbf{w})], \mathbb{E}[1 - f_{-\varphi}(\mathbf{w})]\}$$

Sample $\mathbf{w}_1, \dots, \mathbf{w}_M$ from p($\mathbf{w}$)

$$p(\mathbf{w}) \sim \frac{1}{M} \sum_{i=1}^{M} \delta(\mathbf{w}_i).$$

Then the volume removed by an update $f_\varphi(\mathbf{w})$ c
approximated by:

$$\mathbb{E}[1 - f_\varphi(\mathbf{w})] \simeq \frac{1}{M} \sum_{i=1}^{M} (1 - f_\varphi(\mathbf{w}_i)).$$

Taking one step back

- Part 1) Using Bayes' to update the weights:
  - Metropolis algorithm to actually sample

$$p(\mathbf{w}|I_t) \propto p(\mathbf{w}) \cdot p(I_t|\mathbf{w}).$$

$$\varphi = \Phi(\xi_A) - \Phi(\xi_B)$$

$$f_\varphi(\mathbf{w}) = p(I_t|\mathbf{w}) = \frac{1}{1 + \exp(-I_t \mathbf{w}^\top \varphi)}$$

- Part 2) Synthetically generate pairs of trajectories to give to human:
  - Optimize

$$\max_{x^0, \mathbf{u}_R, \mathbf{u}_H^A, \mathbf{u}_H^B} \quad \min\{\mathbb{E}[1 - f_\varphi(\mathbf{w})], \mathbb{E}[1 - f_{-\varphi}(\mathbf{w})]\}$$

$$\mathbb{E}[1 - f_\varphi(\mathbf{w})] \simeq \frac{1}{M} \sum_{i=1}^{M} (1 - f_\varphi(\mathbf{w}_i)).$$

The Algorithm

Algorithm 1 Preference-Based Learning of Reward Functions

Input: Features ϕ, horizon N, dynamics f, *iter*
Output: Distribution of $\mathbf{w}$: $p(\mathbf{w})$
Initialize $p(\mathbf{w}) \sim \text{Uniform}(B)$, for a unit ball B
While $t < iter$:
  $W \leftarrow M$ samples from $\text{AdaptiveMetropolis}(p(\mathbf{w}))$
  $(x^0, \mathbf{u}_R, \mathbf{u}_H^A, \mathbf{u}_H^B) \leftarrow \text{SynthExps}(W, f)$
  $I_t \leftarrow \text{QueryHuman}(x^0, \mathbf{u}_R, \mathbf{u}_H^A, \mathbf{u}_H^B)$
  $\varphi = \Phi(x^0, \mathbf{u}_R, \mathbf{u}_H^A) - \Phi(x^0, \mathbf{u}_R, \mathbf{u}_H^B)$
  $f_\varphi(\mathbf{w}) = \min(1, I_t \exp(\mathbf{w}^\top \varphi))$
  $p(\mathbf{w}) \leftarrow p(\mathbf{w}) \cdot f_\varphi(\mathbf{w})$
  $t \leftarrow t + 1$
End for

Plan

1. Active Learning Introduction
2. Active Learning and Pairwise Comparison
    a. Active Comparison-Based Learning in Ranking
    b. Active Learning Incorporating the User
3. Active Learning for Reward Functions
    a. Non-Batch methods
    b. Batch method

Previously…

- Trying to pick queries to satisfy:
  - $$\max_{\varphi} \quad \min\{\mathbb{E}[1 - f_{\varphi}(\mathbf{w})], \mathbb{E}[1 - f_{-\varphi}(\mathbf{w})]\}$$
    $$\text{subject to} \quad \varphi \in \mathbb{F}$$

Note this is the same as trying to maximize conditional entropy H(I|w).

Problem: Optimizing each query and waiting takes a long time. What if we batch them? Then the objective becomes:

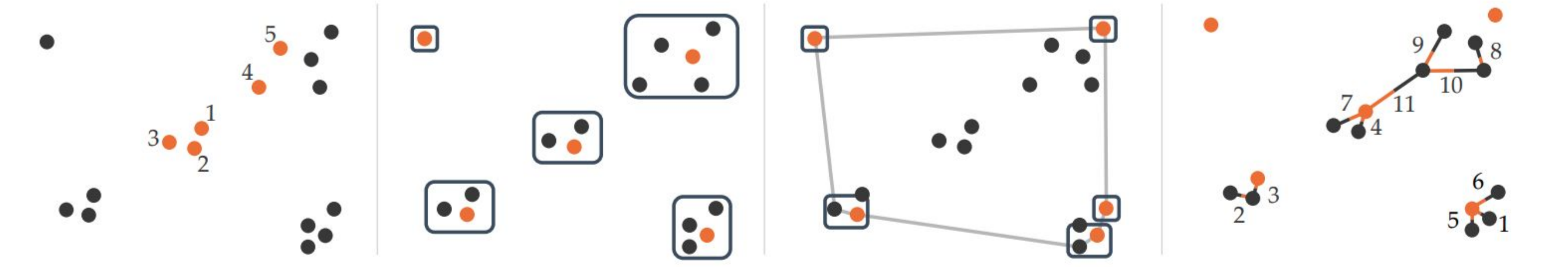
$$\max_{\xi_{ib+1_A}, \xi_{ib+1_B}, \ldots, \xi_{(i+1)b_A}, \xi_{(i+1)b_B}} \mathcal{H}(I_{ib+1}, I_{ib+2}, \ldots, I_{(i+1)b} | \boldsymbol{w})$$

A few approaches

- Greedy:

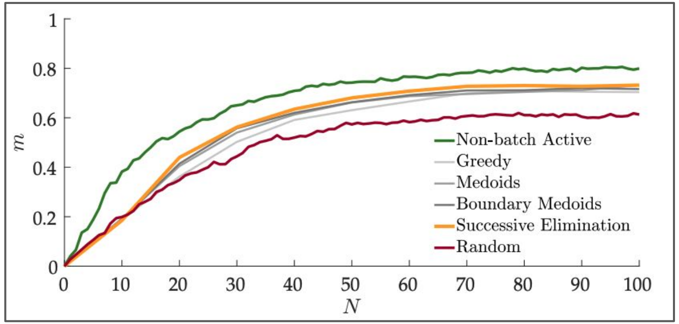
$$\max_{\xi_{ib+1_A},\xi_{ib+1_B}} \mathcal{H}(I_{ib+1}|\boldsymbol{w}) + \cdots + \max_{\xi_{(i+1)b_A},\xi_{(i+1)b_B}} \mathcal{H}(I_{(i+1)b}|\boldsymbol{w})$$

- Medioid Selection: Cluster the B greedy vectors into b < B groups, pick one vector from each group, the medioid.

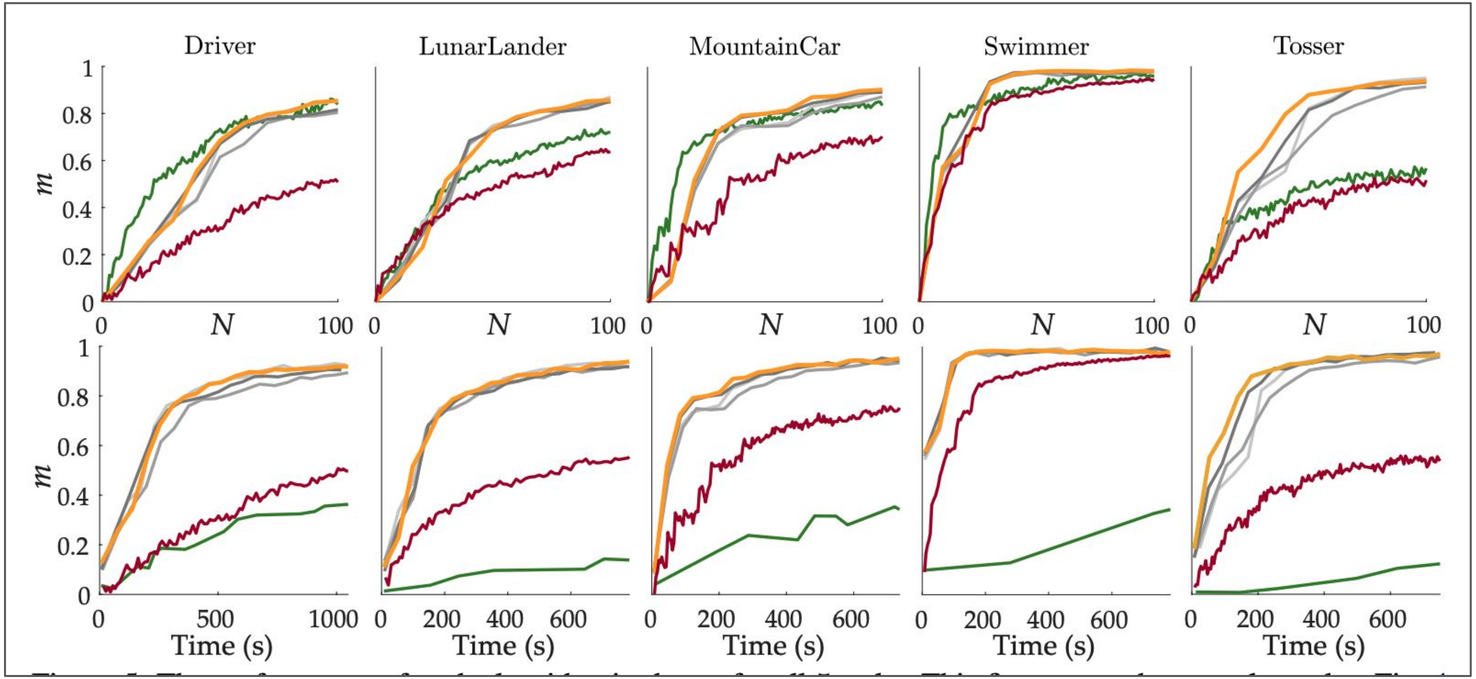


(a) *Greedy Selection.* (b) *Medoids Selection.* (c) *Boundary Medoids Selection.* (d) *Successive Elimination.*

Results



Non-batch Active
Greedy
Medoids
Boundary Medoids
Successive Elimination
Random
m
N
0
10
20
30
40
50
60
70
80
90
100
0.2
0.4
0.6
0.8
1



Driver
LunarLander
MountainCar
Swimmer
Tosser
m
N
Time (s)
0
100
500
1000
200
400
600
0.2
0.4
0.6
0.8
1