Active Learning for Structured Prediction

Rohan Choudhury, Joey Hong, Rohan Doshi

Caltech

May 30, 2017

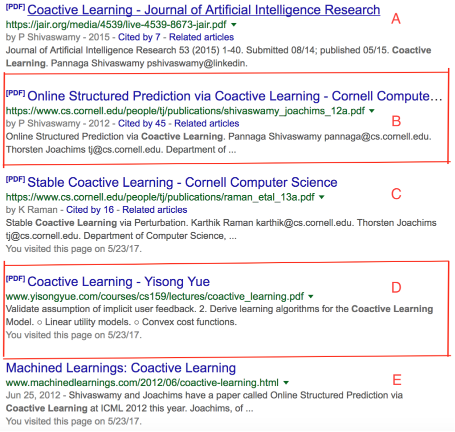
Outline

Outline

Introduction

- Coactive Learning is a model of interaction between learning system and a human user.
- At each step, system predicts an object (possibly structured) given some context, and the user provides slightly improved object as feedback.
- User feedback is often implicit (inferred from behavior).
- The goal of the system is to minimize regret = total deviation from optimal predictions.

Example: Web Search



- User types in a query (e.g. 'coactive learning') to search engine
- Search Engine returns a ranking of documents [A, B, C, D, E, ...]
- User clicks on documents (e.g. B and D)

Introduction

- User feedback is only an incremental improvement, not necessarily optimal
  - For web search, if user clicked B and D, system can infer that the ranking [B, D, A, C, E, ...] would be better.
  - Feedback unlikely the optimal ranking.
- System does not receive optimal prediction, nor any utility functions.

Key Contributions

- Formalized interaction between learning system and user into a Coactive Learning Model.
  - Define regret, and made key modeling assumptions about user feedback via behavior.
- Derive learning algorithms for Coactive Learning Model, including linear utility and convex cost functions.
  - Perform structured output prediction.
  - Show $O(1/\sqrt{T})$ regret bounds.
- Provide empirical evaluations on a movie recommendation and a web-search task.

Outline

Related Work

- The Coactive Learning Model bridges two previously studied forms of feedback
  - Expert Advice Model: Utilities of all possible actions are revealed.
  - Multi-armed Bandit Model: Chooses an action and observe the utility of (only) that action.
- Goal is to minimize regret.

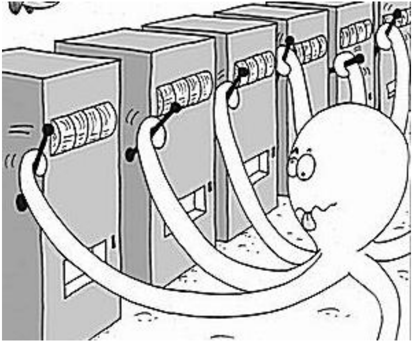


Figure 1: Multi-armed Bandit Problem Illustration

Expert Advice

- Have access to N "experts", who makes predictions $\{f_{i,t}\}$ at time t. Also exists convex loss function ℓ.
- At each time step t:
  - Observes $f_{1,t}, \ldots, f_{N,t}$ and predicts p_t.
  - Outcome y_t is revealed.
  - Suffers loss $\ell(p_t, y_t)$ and experts suffer $\ell(f_{i,t}, y_t)$.
- Try to minimize regret:

$$REG_T = \sum_{i=1}^{T} \ell(p_t, y_t) - \min_{i \in 1 \ldots N} \sum_{i=1}^{T} \ell(f_{i,t}, y_t).$$

- Using Exponential Weighted Average (Multiplicative Weights with continuous labels) algorithm, we can get regret at most $O(\sqrt{T \log N})$.

Multi-armed Bandit Problem

- Set of N "arms" (actions)
- At each time step t:
  - Choose action a_t, with average reward u_i for $1 \leq i \leq N$.
  - Receive reward $X_{i,t}$.
- Denote $u^* = \max_{i \in 1 \ldots K} u_i$. Then, the pseudo-regret is defined as:

$$REG_T = Tu^* - \mathbb{E}\left[\sum_{i=1}^{T} u_t\right].$$

- Key Theme: Exploitation vs Exploration.
  - Exploitation: Choose arm with highest empirical mean reward.
  - Exploration: Test other arms with potentially higher mean reward.

Multi-armed Bandit Problem: UCB1

- UCB1 Algorithm:
  - Play each action j once.
  - For each round t, play the action j maximizing:

$$\bar{x}_j + \sqrt{\frac{2 \log n}{n_j}}$$

  - $\bar{x}_j$ is the average observed reward for j.
  - n_j is the number of times j has been played so far.

Theorem *Suppose UCB1 is run on game with N actions, each with reward $X_{i,t} \in [0, 1]$. The expected regret is at most $O(\sqrt{NT \log T})$.*

Dueling Bandits

- Most closely related to Coactive Learning is the dueling bandits problem.
- Set of N bandits (arms, actions) denoted $\{b_i\}$
- At each time step t:
  - Choose two bandits b_i and b_j to duel.
  - Receive feedback as a stochastic comparison of the bandits, which can be used to construct a pairwise ordering.
- Goal is the find the best bandit b^*.

Dueling Bandits

- Probability that b_i beats b_j in a "duel" depends only on i, j (stationary over time) and is unknown.
  - Probability that b_i beats b_j is $P(b_i > b_j) = \epsilon(b_i, b_j) + 1/2$, with $\epsilon(b_i, b_j) \in (-1/2, 1/2)$.
  - Can be interpreted as fraction of users that prefer b_i to b_j.
  - Duels are independent.
- Regret is defined as:

$$REG_T = \sum_{i=1}^{T} \text{avg}\{\epsilon(b^*, b_i), \epsilon(b^*, b_j)\}.$$

- Can show expected regret of at most $O(\frac{K}{\epsilon_{1,2}} \log T)$, where $\epsilon_{1,2}$ is ϵ between best and second best bandit.
- Difference between dueling bandits and Coactive Learning is that only b_i is given to user, and feedback determines b_j, which is guaranteed to be better.

Summary of Problems

- Play for T rounds.
- Set of N possible arms/actions.

Problem	Action	Feedback
Expert Advice	Chooses arm (expert)	Reward for every ar
Multi-armed Bandit	Chooses arm	Reward for chosen
Dueling Bandit	Chooses two arms (bandits)	The better arm
Coactive Learning	Chooses arm	Any better arm

Table 1: Summary of Related Problems.

Outline

Coactive Learning Model

- Want to model (in rounds) interaction between learning system and user, where both system and user want to obtain good results
- At reach round t:
  - System observes context $\mathbf{x}_t \in \mathcal{X}$.
  - Presents a structured object $\mathbf{y}_t \in \mathcal{Y}$.
  - User returns an improved object $\bar{\mathbf{y}}_t \in \mathcal{Y}$
- The utility of $\mathbf{y}_t \in \mathcal{Y}$ to the user for context $\mathbf{x}_t \in \mathcal{X}$ is described by utility function $U(\mathbf{x}_t, \mathbf{y}_t)$.
- Improved object $\bar{\mathbf{y}}_t$ satisfies,

$$U(\mathbf{x}_t, \bar{\mathbf{y}}_t) > U(\mathbf{x}_t, \mathbf{y}_t)$$

Coactive Learning Model

- User performs an approximate utility-maximizing search over some user-defined subset $\bar{\mathcal{Y}}_t$ of all possible $\mathcal{Y}$.
- User is only approximately rational, however, so $\bar{\mathbf{y}}_t$ is typically not the optimal label

$$\mathbf{y}_t^* := \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} U(\mathbf{x}_t, \mathbf{y})$$

- User is assume to provide reliable preference feedback. However, doesn't know the cardinal utility U when generating feedback.
- Very different from supervised learning approaches which require $(\mathbf{x}_t, \mathbf{y}_t^*)$.

Coactive Learning Model

- Aim of algorithm is to present objects with utility close to $\mathbf{y}_t^*$.
- Whenever, the algorithm presents an object $\mathbf{y}_t$ under context $\mathbf{x}_t$, we say that it suffers a regret $U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)$ at time step t.
- Consider average regret suffered over T steps,

$$REG_T = \frac{1}{T} \sum_{t=1}^{T} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right)$$

- Goal is to minimize REG_T. Note: Real value of U is never observed by the learning algorithm, but only semi-revealed by preferences.

Quantifying Feedback Quality

- Quantify feedback quality by how much improvement $\bar{\mathbf{y}}$ provides in utility space.
- Say that user feedback is *strictly α-informative* when the following inequality is satisfied:

$$U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \geq \alpha(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t))$$

  for some $\alpha \in (0, 1]$.
- Means that utility of $\bar{\mathbf{y}}_t$ is higher than $\mathbf{y}_t$ by some fraction α of maximum difference.

Quantifying Feedback Quality

- Feedback is α-*informative* once we introduce slack variables:

$$U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \geq \alpha(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)) - \xi_t$$

- Even weaker: feedback is *expected* α-*informative* if expectation achieves positive utility gain:

$$\mathbf{E}_t[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] \geq \alpha(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)) - \bar{\xi}_t$$

- Expectation is over the user's choice of $\bar{\mathbf{y}}_t$ given $\mathbf{y}_t$ under context $\mathbf{x}_t$ (i.e. distribution $\mathbf{P}_{\mathbf{x}_t}[\bar{\mathbf{y}}_t|\mathbf{y}_t]$)

Study: User Clicks

- Experimentally validate that user behavior implies reliable preferences.
- Subjects (16 undergraduate students) were asked to answer 10 questions (5 informational, 5 navigational) using the Google search engine.
- Used following strategy to infer rankings $\bar{\mathbf{y}}$:
  - Prepend to ranking $\mathbf{y}$ for each query all results that the user clicked.
  - If Google gave rankings [A, B, C, D, ...], and user clicks B and D, then inferred ranking becomes [B, D, A, C, ...].

Study: User Clicks

- Measure utility in terms of retrieval quality from Information Retrieval

$$DCG@10(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{10} \frac{r(\mathbf{x}, \mathbf{y}[i])}{\log i + 1}$$

- $r(\mathbf{x}, \mathbf{y}[i]) \in [0...5]$ is the normalized relevance score of the i-th document (ground truth assessed by human assessors).
- Want that feedback ranking $\bar{\mathbf{y}}$ better than rankings $\mathbf{y}$

$$DCG@10(\mathbf{x}, \bar{\mathbf{y}}) > DCG@10(\mathbf{x}, \mathbf{y})$$

Study: User Clicks

- Had to confirm that quality of feedback was not affected by quality of current prediction.
- 3 User Groups (each $\approx 1/3$ of entire sample) received prediction in 3 different orderings:
  - Normal: Top 10 results in normal order.
  - Reverse: Top 10 results in reverse order.
  - Swapped: Top 2 results are swapped

Study: User Clicks

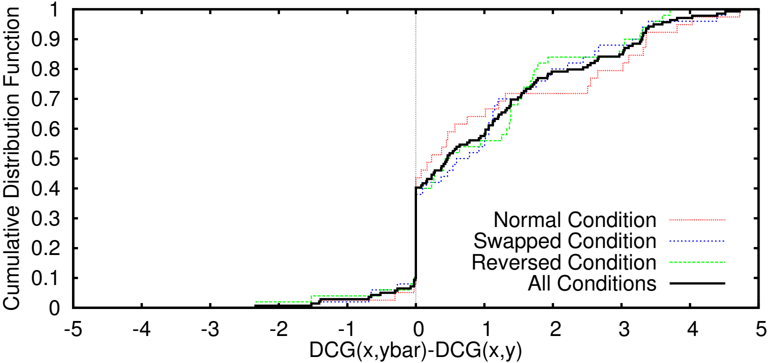


Figure 2: Cumulative distribution of utility differences

- CDF shifted right of 0 implies that implicit feedback improves utility.

Outline

Coactive Learning Algorithms

- Model utility function with linear model

$$U(\mathbf{x}, \mathbf{y}) = \mathbf{w}_*^\top \phi(\mathbf{x}, \mathbf{y})$$

- $\mathbf{w}_* \in \mathbf{R}^N$ is an unknown parameter vector
- $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbf{R}^N$ is a joint feature map.
  - If $\mathbf{x}$ were queries, and $\mathbf{y}$ rankings, then joint features could include relevancy of documents.
- Want $||\phi(\mathbf{x}, \mathbf{y})||_{\ell_2} \leq R$ for any $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$.

Preference Perceptron

Algorithm 1 Preference Perceptron.

Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  Observe $\mathbf{x}_t$
  Present $\mathbf{y}_t \leftarrow \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$
  Obtain feedback $\bar{\mathbf{y}}_t$
  Update: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$
end for

- Maintains a weight vector $\mathbf{w}_t$ which is initialized to $\mathbf{0}$
- In each time step t, updates weight vector $\mathbf{w}_t$ in the direction $\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$.

Preference Perceptron - Bounds

Theorem *The average regret of the preference perceptron algorithm can be upper bounded, for any* $\alpha \in (0, 1]$ *and for any* $\mathbf{w}_*$ *as follows*:

$$REG_T \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R||\mathbf{w}_*||}{\alpha \sqrt{T}}.$$

- Recall: Feedback is α-*informative*

$$U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \geq \alpha(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)) - \bar{\xi}_t$$

- Recall: $||\phi(\mathbf{x}, \mathbf{y})||_{\ell_2} \leq R$

Preference Perceptron - Proof

First, prove $||\mathbf{w}_{T+1}||^2 \leq 4RT^2$.

$$\begin{aligned}
\mathbf{w}_{T+1}^\top \mathbf{w}_{T+1} &= \mathbf{w}_T^\top \mathbf{w}_T + 2\mathbf{w}_T^\top(\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)) \\
&\quad + (\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T))^\top(\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)) \\
&\leq \mathbf{w}_T^\top \mathbf{w}_T + 4R^2 \\
&\leq 4R^2 T
\end{aligned}$$

- First line is application of update rule.
- Second line is from choice of $\mathbf{y}_t = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$, and $||\phi(\mathbf{x}, \mathbf{y})|| \leq R$
- Third line is from repeated application of inequality, starting from $\mathbf{w}_1^\top \mathbf{w}_1 = 0$.

Preference Perceptron - Proof

Next, bound $\sum_{t=1}^{T} \left(U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)\right)$.

$$\begin{aligned}\sum_{t=1}^{T} \left(U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)\right) = \mathbf{w}_{T+1}^\top \mathbf{w}_* &\leq ||\mathbf{w}_{T+1}|| \, ||\mathbf{w}_*|| \\ &\leq 2R\sqrt{T}||\mathbf{w}_*||.\end{aligned}$$

- Observe the property that

$$\begin{aligned}\mathbf{w}_{T+1}^\top \mathbf{w}_* &= \mathbf{w}_T^\top \mathbf{w}_* + \left(\phi(\mathbf{x}_T, \bar{\mathbf{y}}_T) - \phi(\mathbf{x}_T, \mathbf{y}_T)\right)^\top \mathbf{w}_* \\ &= \sum_{t=1}^{T} \left(U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)\right)\end{aligned}$$

- First inequality from Cauchy-Schwarz.

Preference Perceptron - Proof

Bound REG_T.

$$\alpha \sum_{t=1}^{T} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right) - \sum_{t=1}^{T} \xi_t \leq 2R\sqrt{T} ||\mathbf{w}_*||,$$

$\square$

- Assume that α-informative model of feedback.
- If the user feedback is strictly α-informative, then all slack variables vanish and $REGT = O(1/\sqrt{T})$.

Preference Perceptron - Lower Bound

Lemma *For any coactive learning algorithm $\mathcal{A}$ with linear utility, there exist $\mathbf{x}_t$, objects $\mathcal{Y}$ and $\mathbf{w}_*$ such that REG_T of $\mathcal{A}$ in T steps is $\Omega(1/\sqrt{T})$.*

- Consider $\mathcal{Y} = \{-1, +1\}$, $\mathcal{X} = \{\mathbf{x} \in \mathbf{R}^T : ||\mathbf{x}|| = 1\}$
- Define joint feature map $\phi(\mathbf{x}, \mathbf{y}) = \mathbf{yx}$
- Consider T contexts $\mathbf{e}_1, \ldots, \mathbf{e}_T$, with each $\mathbf{e}_i$ standard i-th basis vector. Let $\mathbf{y}_1, \ldots \mathbf{y}_T$ be the sequence of outputs.
- Let $\mathbf{w}_* = [-\mathbf{y}_1/\sqrt{T} \ -\mathbf{y}_2/\sqrt{T} \cdots -\mathbf{y}_T/\sqrt{T}]^\top$. Notice $||\mathbf{w}_*|| = 1$.
- Let the user feedback on the t^{th} step be $-\mathbf{y}_t$ (always α-informative with $\alpha = 1$).
- Regret is $\frac{1}{T}\sum_{t=1}^{T}(\mathbf{w}_*^\top \phi(\mathbf{e}_t, \mathbf{y}_t^*) - \mathbf{w}_*^\top \phi(\mathbf{e}_t, \mathbf{y}_t)) = \Omega(\frac{1}{\sqrt{T}})$

$\square$

Preference Perceptron - Batch Update

- Sometimes, there are too high volumes of feedback to update every round.
- Perform variant of algorithm that makes update every k iterations. Uses $\mathbf{w}_t$ obtained from the previous update until the next update.
- Can show regret bound:

$$REG_T \leq \frac{1}{\alpha T} \sum_{t=1}^{T} \xi_t + \frac{2R||\mathbf{w}_*||\sqrt{k}}{\alpha \sqrt{T}}$$

Preference Perceptron - Expected α-Informative Feedback

- If we only want a bound on expected regret, we can use a weaker Expected α-Informative Feedback.
- Can show regret bound

$$\mathbf{E}[REG_T] \leq \frac{1}{\alpha T}\sum_{t=1}^{T}\bar{\xi}_t + \frac{2R||\mathbf{w}_*||}{\alpha\sqrt{T}}.$$

- Take expectations over user feedback to get:

$$\mathbf{E}[\mathbf{w}_{T+1}^\top \mathbf{w}_{T+1}] \leq 4R^2 T.$$

Rest of proof follows from application of Jensen's inequality.

Convex Loss Minimization

- Can generalize results to minimize convex losses defined on linear utility differences
- At every time step: there is an (unknown) convex loss function $c_t : \mathbf{R} \to \mathbf{R}$ which determines the loss $c_t(U(\mathbf{x}_t, \mathbf{y}_t) - U(\mathbf{x}_t, \mathbf{y}_t^*))$ at time t.
  - Functions c_t are assumed to be non-increasing.
  - Sub-derivatives of the c_t's are assumed to be bounded (i.e., $c_t'(\theta) \in [-G, 0]$ for all t and for all $\theta \in \mathbf{R}$)
- The vector $\mathbf{w}_*$ is assumed from a closed and bounded convex set $\mathcal{B}$ with diameter $|\mathcal{B}|$.

Convex Preference Perceptron

Algorithm 2 Convex Preference Perceptron.

Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  Set $\eta_t \leftarrow \frac{1}{\sqrt{t}}$
  Observe $\mathbf{x}_t$
  Present $\mathbf{y}_t \leftarrow \text{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y})$
  Obtain feedback $\bar{\mathbf{y}}_t$
  Update: $\bar{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t + \eta_t G(\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t))$
  Project: $\mathbf{w}_{t+1} \leftarrow \arg\min_{\mathbf{u} \in \mathcal{B}} \|\mathbf{u} - \bar{\mathbf{w}}_{t+1}\|^2$
end for

- Introduces rate η_t associated with the update at time t
- After every update, the resulting vector $\bar{\mathbf{w}}_{t+1}$ is projected back to the set $\mathcal{B}$.

Convex Preference Perceptron

- Algorithm minimizes average convex loss. We have bound

$$\frac{1}{T}\sum_{t=1}^{T} c_t(U(\mathbf{x}_t,\mathbf{y}_t) - U(\mathbf{x}_t,\mathbf{y}_t^*))$$

$$\leq \frac{1}{T}\sum_{t=1}^{T} c_t\left(0\right) + \frac{2G}{\alpha T}\sum_{t=1}^{T}\xi_t + \frac{1}{\alpha}\left(\frac{|\mathcal{B}|G}{2\sqrt{T}} + \frac{|\mathcal{B}|G}{T} + \frac{4R^2 G}{\sqrt{T}}\right).$$

- $c_t(0)$ is the minimum possible convex loss
- Under strict α-informative feedback, average loss approaches $\mathcal{O}(1/\sqrt{T})$.

Outline

Structured Feedback: Learning to Rank

- STRONG VS. WEAK FEEDBACK: See how the regret of the Preference Perceptron algortihm changes with feedback quality
- Feedback of different qualities α was received in the following way:

  Given a predicted ranking y_t a user would go down the list and find five URLs. It is a requirement that when these URLs are placed at the top of the list the resulting $\bar{\mathbf{y}}_t$ satisfied the strictly α-informative feedback condition w.r.t. the optimal $\mathbf{w}_*$.

Structured Feedback: Learning to Rank

- The Preference Perceptron algorithm was used on the Yahoo! learning to rank dataset (Chapelle & Chang, 2011). It consists of query-url feature vectors (denoted as $\mathbf{x}_i^q$ for query q and URL i), each with a relevance rating r_i^q that ranges from 0 (irrelevant) to 4 (perfectly relevant).
- The joint feature map was defined as follows:

$$\phi(q, \mathbf{y}) = \sum_{i=1}^{5} \frac{\mathbf{x}_{\mathbf{y}_i}^q}{\log(i+1)}$$

- y denotes a ranking such that y_i is the index of the URL which is placed at position i in the ranking. This measure considers the top five URLs for a query q and computes a score based on a graded relevance

Structured Feedback: Learning to Rank

- For query q_t at time step t, the Preference Perceptron algorithm presents the ranking $\mathbf{y}_t^q$ that maximizes $\mathbf{w}^T \phi(q_t, \mathbf{y})$
- The utility regret is given by

$$\frac{1}{T} \sum_{i=1}^{T} \mathbf{w}_*^T (\phi(q_t, \mathbf{y}^{q_t *}) - \phi(q_t, \mathbf{y}^{q_t}))$$

Structured Feedback: Learning to Rank

- RESULTS:

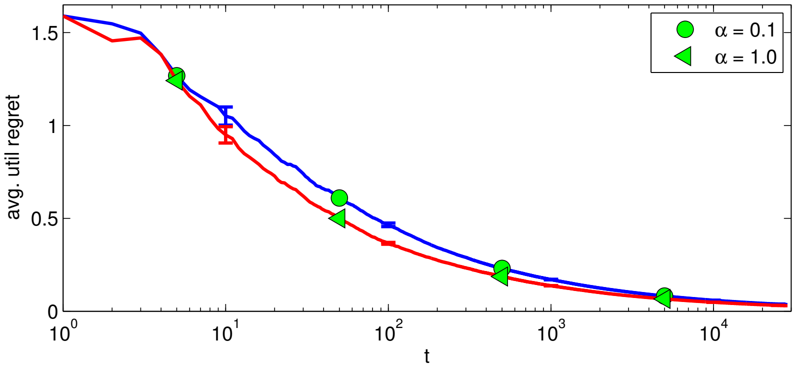


Figure 3: Regret based on strictly α-informative feedback

Structured Feedback: Learning to Rank

- RESULTS:
- The regret with respect to $\alpha = 1.0$ smaller than the regret with respect to $\alpha = 0.1$.
- The regret approaches zero since there is no noise for both αs
- However, the difference in regret is much less than a factor of ten. This can be explained by the fact that feedback was stronger than expected.

Structured Feedback: Learning to Rank

- NOISY FEEDBACK: See how the preference perceptron algorithm performs on noisy feedback
- Used the actual relevance labels provided in the Yahoo! dataset for user feedback. Now, given a ranking for a query, the user would go down the list inspecting the top 10 URLs (or all the URLs if the list is shorter) as before. Five URLs with the highest relevance labels (r_i^q) are placed at the top five locations in the user feedback.
- This produces noisy feedback since no linear model can perfectly fit the relevance labels on this dataset.

Structured Feedback: Learning to Rank

- RESULTS

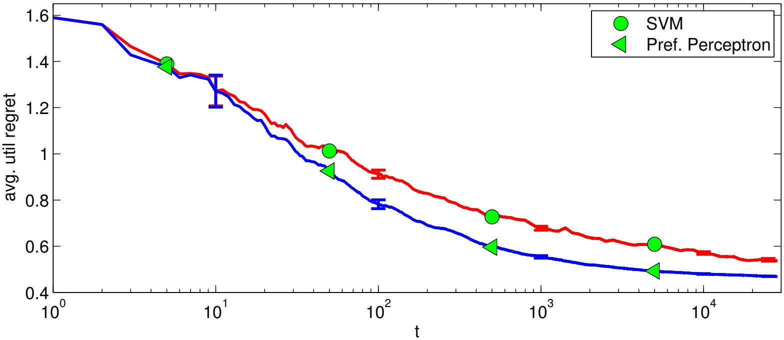


Figure 4: Regret vs time based on noisy feedback

Item Feedback: Movie Recommendation

- GOAL: Evaluate the preference perceptron on the atomic prediction task of movie recommendation.
- Used the MovieLens dataset, which contained a million ratings over 3090 movies rated by 6040 users.
- Users were divided into two sets. The first was used to obtain a feature vector $\mathbf{m}_j$ for each movie using SVD embedding for collaborative filtering (Bell and Koren, 2007). The dimensionality of these feature vectors and the regularization parameters were chosen to optimize cross-validation accuracy

Item Feedback: Movie Recommendation

- The feature vectors $\mathbf{m}_j$ were used to recommend movies to the second set of users. For each user i in the second set a best least squares approximation $\mathbf{w}_{i*}^T \mathbf{m}_j$ was found for the users utility functions on the available ratings. This allows us to compute the utility values for movies that were not rated by user i.
- We can then also measure regret as

$$\frac{1}{T} \sum_{i=1}^{T} \mathbf{w}_*^T (\mathbf{m}_{t*} - \mathbf{m}_t)$$

where $\mathbf{m}_{t*}$ is the best available movie and $\mathbf{m}_t$ is the recommended movie

Item Feedback: Movie Recommendation

- STRONG VS. WEAK FEEDBACK: Explore how the performance of the Preference Perceptron changes with feedback quality α
- A movie with maximum utility based on the current w_t of the algorithm was recommended, and the user returns as feedback a movie with the smallest utility that still satisfied α informative feedback according to w_{i*}.
- This was done for every user in the second set for 1500 iterations.
- Regret was calculated separately for each user and then all regrets over all users were averaged.

Item Feedback: Movie Recommendation

- RESULTS:

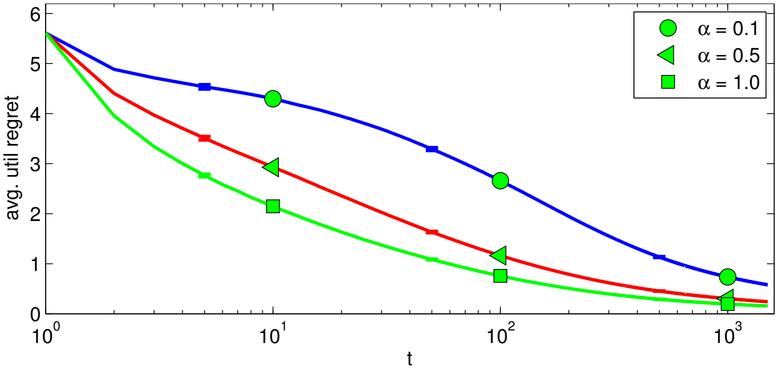


Figure 5: Regret based on strictly α-informative feedback

Item Feedback: Movie Recommendation

- NOISY FEEDBACK: Evaluate Preference Perceptron performance when the user feedback does not match the linear utility model used by the algorithm.
- Feedback is given based on the actual ratings when available. In every iteration, the user returned a movie with one rating higher than the one presented to her. If the algorithm already presented a movie with the highest rating, it was assumed that the user gave the same movie as feedback.

Item Feedback: Movie Recommendation

- RESULTS:

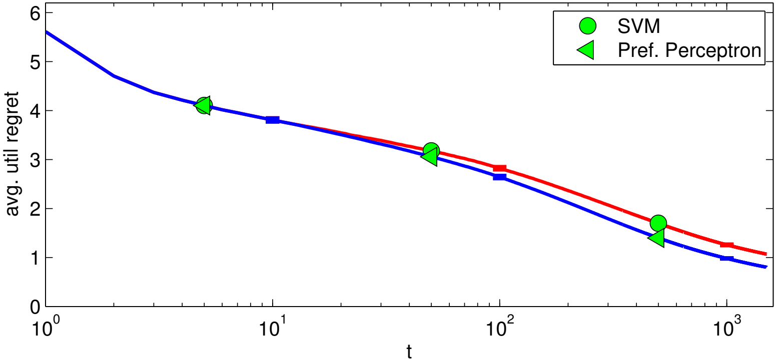


Figure 6: Regret based on noisy feedback

Outline

Motivation

- In supervised learning, obtaining labels can be expensive.
- What is the fewest number of labels we can have to still achieve good results?

Active vs. Passive

- In *passive* learning, simply train on labeled examples
- In *active* learning, the system can request labels for some examples.
- Consider example of learning a 1D threshold:
  - An active learner will need logarithmically many samples as a passive learner, since he can pick points based on some variant of binary search.
- In above example, active learner achieves exponential advantage over passive counterpart.

Active vs. Passive: 1D Threshold

- We want to determine a threshold $T \in [0, 1]$ based on given examples.

Figure 7: 1D threshold problem.

- With supervised learning, number of examples needed to learn within ϵ error is $O(\frac{1}{\epsilon})$
- With active learning, get $O(\log\left(\frac{1}{\epsilon}\right))$. (can do a binary search method)

Active Learning: Sampling Methods

- Learning system makes queries regarding an unlabeled training example to an oracle, which then labels it for us.
- Stream-based Sampling:
  - Receives training examples from a stream of data
  - System chooses whether or not to ask the oracle to label the sample.
- Pool-based Sampling:
  - Small amount of labeled training examples and a large amount of unlabeled training examples.
  - Look at entire pool of training data to choose which to query and label.
  - Pick which to label via some greedy metric (uncertainty)
- Second paper deals with pool-based sampling.

Outline

Introduction

- Luo et al, Latent Structured Active Learning (2013)
- This paper is about applying active learning to general structured prediction.
- Often when we predict structures, we need a lot of labels. Active learning can solve this problem!

Key Contributions

- Previous work has focused on active learning cases with exact inference.
- This paper gives approximate approaches for general graphical models
- Provides general algorithm for efficient latent structured prediction
- Provides two algorithms for active learning-based latent structured prediction
- Demonstrates algorithms in 3D room layout prediction - requires 10% of the labels

Outline

Max-Likelihood Structured Prediction

- Let $\mathcal{X}$ be the input space, with the corresponding structured labeled space being $\mathcal{S}$.
- Define $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^F$ to be the joint feature map to an F-dimensional feature space.
- Let w be an F-dimensional weight vector.
- For an input x with label s, let the *scoring function* be

$$w^T \phi(x, s)$$

Max-Likelihood Structured Prediction

- Furthermore, we'll consider the case where

$$p_w(s \mid x) \propto \exp(\mathbf{w}^T \phi(x, s))$$

  with w being an F-dimensional weight vector, and $x \in \mathcal{X}, s \in \mathcal{S}$.
- We want $\mathbf{w}$ such that if $s \in \mathcal{S}$ is a good label for $x \in \mathcal{X}$, it has a high score $\mathbf{w}^T \phi(x, s)$.

Supervised Setting

- We have a dataset $\mathcal{D} = \{(x_i, s_i)_{i=1}^N\}$
- We have a *task-loss* function $l_{(x,s)}(\hat{s})$ for an estimate $\hat{s}$. This describes the "fitness" of an estimate and imposes structure on our output.
- Using this, consider *loss-augmented distribution*

$$p_{(x,s)}(s \mid w) \propto \exp(\mathbf{w}^T \phi(x, s) + l_{(x,y)}(s))$$

- Let's break this down:
  - Higher score means more probable
  - Place more mass on estimates with high loss (makes the task more difficult)

Supervised Setting

- We want to *minimize* the *negative log-likelihood*.
- This is given by

$$-L(s; x, w) = -\ln\left(p(w)\prod_{(x,s)\in\mathcal{D}} p_{(x,s)}(s \mid w)\right)$$

- Use $p(w) \propto exp(-|w|_p^p)$ as a prior on the parameters.
- Define the p-norm for $\mathbf{w} = (w_1, w_2, ...w_F)$ to be

$$||\mathbf{w}||_p^p = \left(\sum_{i=1}^{F} w_i^p\right)^{1/p}$$

- You get to choose p. For example, $p = 2$ would correspond to the L2 norm.

Supervised Setting

- Plugging it all in, we get a cost function

$$\frac{C}{p}||w||_p^p +$$

$$\sum_{(x,y)\in\mathcal{D}} \left(\epsilon \ln \sum_{\hat{s}\in\mathcal{S}} \exp\left(\frac{\mathbf{w}^T \phi(x,\hat{s}) + l_{(x,y)}(\hat{s})}{\epsilon} \right) - \mathbf{w}^T \phi(x,s) \right)$$

- Put in an ϵ as a temperature term.
  - $\epsilon \to 0$ makes it a simple max
  - $\epsilon \to 1$ makes it a normal log likelihood
- This is convex, but has a sum over exponentially many possibilities $\hat{s}$
- There are various ways to solve this problem, covered in depth in previous lectures

Dealing with Latent Variables

- What are latent variables?
- Formally, given $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}\}$ we say each pair has $x \in \mathcal{X}$ and *partial label data* $y \in \mathcal{Y} \subset \mathcal{S}$.
- In the latent variable setting, we assume the label space is of the form $\mathcal{S} = \mathcal{Y} \times \mathcal{H}$ with $\mathcal{Y}, \mathcal{H}$ being non-intersecting subspaces of $\mathcal{S}$. $\mathcal{H}$ represents the "labels" for the *latent variables*.

Latent Variables

- We need to set up our likelihood and objective like before.
- Recall the task-loss function $l_{(x,y)}(\hat{s})$. Then, our loss-augmented distribution is

$$p_{(x,y)}(\hat{y} \mid \mathbf{w}) \propto \sum_{h \in \mathcal{H}} p_{(x,y)}(\hat{y}, \hat{h} \mid \mathbf{w}) = \sum_{h \in \mathcal{H}} p_{(x,y)}(\hat{s} \mid \mathbf{w})$$

Latent Variables

Using the distribution to compute the likelihood, we get

$$\frac{C}{p}||\mathbf{w}||_p^p + \sum_{(x,y)\in\mathcal{D}} \left(\epsilon \ln \sum_{\hat{s}\in S} \exp\left(\frac{\mathbf{w}^T\phi(x,\hat{s}) + l_{(x,y)}(\hat{s})}{\epsilon} \right) \right.$$
$$\left. - \epsilon \sum_{\hat{h}\in\mathcal{H}} \exp\left(\frac{\mathbf{w}^t\phi(x,y,\hat{h}) + l^c_{(x,y)}(y,\hat{h})}{\epsilon} \right) \right)$$

- We have different task loss functions for labeled and partially labeled.
- This is not convex.

Latent Variables

- How can we solve this objective?
- Follow Yuille & Rangarajan (2003) and upper bound the concave part (the second term in the sum)
  - Achieve upper bound via minimization over dual variables
- Will allow us to do an approximation with an Expectation-Maximization (EM) approach rather than exact inference

Latent Variables

- To make the minimization more tractable, use a joint distribution $q_{(x,y)}$.
- Also, we can often decompose the feature vector:

$$\phi_k(x, s) = \sum_{i \in V_{k,x}} \phi_{k,i}(x, s_i) + \sum_{\alpha \in E_{k,x}} \phi_{k,a}(x, s_\alpha)$$

- Here, $V_{k,x}$ are the unary potentials for the feature k (i.e, weak interaction in the graph)
- $\alpha \in E_{k,x}$ is a high-order variable interaction set in the k-th feature.

Latent Variables

Claim The function

$$\frac{C}{p}||\mathbf{w}||_p^p + \sum_{(x,y)\in\mathcal{D}} \Bigg(\epsilon \ln \sum_{\hat{s}\in S} \exp\left(\frac{\mathbf{w}^T\phi(x,\hat{s}) + l_{(x,y)}(\hat{s})}{\epsilon} \right) \\ - \epsilon H(q_{(x,y)}) - \mathbb{E}_{q_{(x,y)}}[\mathbf{w}^T\phi(x,(y,\hat{h})) + l^c(x,(y,\hat{h})] \Bigg)$$

which is convex in $\mathbf{w}$ and $q_{(x,y)}$ separately is an upper bound on the previous cost function, $\forall q_{(x,y)} \in$ the probability simplex Δ, H the entropy.
Proof omitted

Latent Structured Prediction

Theorem The approximation of the program minimizing the previous equation takes the form

Program 1 *Approximated structured prediction with latent variables*

$$
\min_{d,\lambda,w} \; f_1 \left\{ \begin{aligned} & \frac{C}{2}\|w\|_2^2 + \sum_{(x,y)\in\mathcal{D}} \left(\sum_{i\in\mathbb{S}} \epsilon c_i \ln \sum_{s_i} \exp\left(\frac{\phi_{(x,y),i}(s_i) - \sum_{\alpha\in N(i)} \lambda_{(x,y),i\to\alpha}(s_i)}{\epsilon c_i} \right) + \right. \\ & \qquad \left. + \sum_{\alpha\in E} \epsilon c_\alpha \ln \sum_{s_\alpha} \exp\left(\frac{\phi_{(x,y),\alpha}(s_\alpha) + \sum_{i\in N(\alpha)} \lambda_{(x,y),i\to\alpha}(s_i)}{\epsilon c_\alpha} \right) \right) - \end{aligned} \right.
$$

$$
f_2 \left\{ \; -\sum_r w_r \left(\sum_{(x,y)} \left(\sum_{i\in\mathbb{Y}} \phi_{r,i}(x,y_i) + \sum_{i\in\mathbb{H},h_i} \phi_{r,i}(x,h_i) d_{(x,y),i}(h_i) + \sum_{\alpha\in E,h_\alpha} \phi_{r,\alpha}(x,(y,h)_\alpha) d_{(x,y),\alpha}(h_\alpha) \right) \right) \right.
$$

$$
f_3 \left\{ \begin{aligned} & -\sum_{(x,y)} \left(\sum_{i\in\mathbb{H},h_i} \ell^c_{(x,y),i}(x,h_i) d_{(x,y),i}(h_i) + \sum_{\alpha\in E_{\mathbb{H}},h_\alpha} \ell^c_{(x,y),\alpha}(x,(y,h)_\alpha) d_{(x,y),\alpha}(h_\alpha) \right) \\ & -\sum_{(x,y)} \left(\sum_{i\in\mathbb{H}} \epsilon\hat{c}_i H(d_{(x,y),i}) + \sum_{\alpha\in E_{\mathbb{H}}} \epsilon\hat{c}_\alpha H(d_{(x,y),\alpha}) \right) \end{aligned} \right.
$$

$$
\text{s.t.} \quad \left. \begin{aligned} & \sum_{h_\alpha \setminus h_i} d_{(x,y),\alpha}(h_\alpha) = d_{(x,y),i}(h_i) \quad \forall (x,y), i\in\mathbb{H}, \alpha\in N(i), h_i\in\mathcal{S}_i \\ & d_{(x,y),i}, d_{(x,y),\alpha} \in \Delta \end{aligned} \right\} := d_{(x,y)} \in \mathcal{C}_{(x,y)} \quad \forall (x,y)\in\mathcal{D}
$$

Generic Latent Structured Prediction Algorithm

Algorithm 1 latent structured prediction

Input: data $\mathcal{D}$, initial weights w
repeat
  repeat
    //solve latent variable prediction problem
    $\min_d f_2 + f_3$ s.t. $\forall(x,y)\ d_{(x,y)} \in \mathcal{D}_{(x,y)}$
  until convergence
  //message passing update
  $\forall(x,y), i \in \mathbb{S} \quad \lambda_{(x,y),i} \leftarrow \nabla_{\lambda_{(x,y),i}}(f_1 + f_2) = 0$
  //gradient step with step size η
  $w \leftarrow w - \eta \nabla_w (f_1 + f_2)$
until convergence
Output: weights w, beliefs d

- Using the previous theorem, can solve the objective with EM/CCCP (concave-convex)

Quick Summary

- In supervised setting, structured prediction understood
- In supervised latent variable setting, we have an algorithm for structured prediction

Outline

Active Learning Algorithms

- In the active learning setting, we have a partially labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N_L}$ with $x_i \in \mathcal{X}, y_i \in \mathcal{Y} \subset \mathcal{S}$.
- We assume latent variable structure, so $\mathcal{S} = \mathcal{Y} \times \mathcal{H}$.The information $h \in H$ is the latent part.
- We also have an unlabeled set $\mathcal{D}_{\mathcal{U}} = \{(x_i)_{i=1}^{N_u}\}$.

Quantifying Uncertainty

- We need some measure of uncertainty to determine which point to query.
- Entropy given by

$$H(d_i) = -\sum_{h_i=1}^{|\mathcal{H}_i|} d_i(h_i) \log d_i(h_i)$$

- d_is are the local beliefes

Active Learning Algorithms

- What part of the graph should we label to get the best model with the least supervision?
- Select random variable in graph to label based on local entropies.
- This is a measure of uncertainty in parts of the graph.
- Idea is to get labels for random variables of highest uncertainty, update model, then query again.

Separate Active Algorithm

Algorithm 2 Separate active

Input: data $\mathcal{D}_S, \mathcal{D}_U$, initial weights w
repeat
  $(w, d_S) \leftarrow$ Alg. $1(\mathcal{D}_S, w)$
  $d_U \leftarrow$ Inference$(\mathcal{D}_U)$
  $i^* \leftarrow \arg\max_i H(d_i)$
  $\mathcal{D}_S \leftarrow \mathcal{D}_S \cup \{(x_{i^*}, y_{i^*})\}, \mathcal{D}_U \leftarrow \mathcal{D}_U \setminus x_{i^*}$
until sufficiently certain
Output: weights w

- Learns parameters based on labeled data first
- Then, perform inference of unlabeled data to find next random variable to label, adds to labeled

Joint Active Algorithm

Algorithm 3 Joint active

Input: data $\mathcal{D}_S$, $\mathcal{D}_U$, initial weights w
repeat
  $(w, d) \leftarrow \text{Alg. } 1(\mathcal{D}_S \cup \mathcal{D}_U, w)$
  $i^* \leftarrow \arg\max_i H(d_i)$
  $\mathcal{D}_S \leftarrow \mathcal{D}_S \cup \{(x_{i^*}, y_{i^*})\}, \mathcal{D}_U \leftarrow \mathcal{D}_U \setminus x_{i^*}$
until sufficiently certain
Output: weights w

- Learns parameters based on labeled *and* unlabeled data
- Then, perform inference of unlabeled data to find next random variable to label

How This Works

- Recall we have a closed form for entropy

$$H(d_i) = -\sum_{h_i=1}^{|\mathcal{H}_i|} d_i(h_i) \log d_i(h_i)$$

- Compute over all random variables, pick the least certain one to label (highest entropy)
- Separate vs. Active:
  - Separate only learns based on labeled examples, then does inference to get local beliefs
  - Joint learns over all examples, so has local beliefs for all random variable - can be more expensive

Outline

Experimental Evaluation

- TASK: Predict the 3D layout of rooms from a single image
- Using the Manhattan world assumption (the existence of three dominant vanishing points which are orthonormal), and given the vanishing points, this problem can be formulated as inference in a pairwise graphical model composed of four random variables.
- Performance is measured as the percentage of pixels that have been correctly labeled as, left-wall, right-wall, front-wall, ceiling or floor.

Experimental Evaluation

- RESULTS:

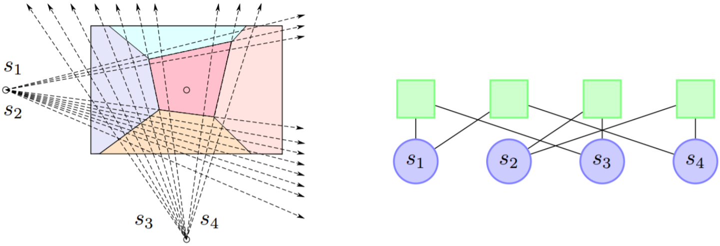


Figure 8: Parameterization and factor graph for the 3D layout prediction task.

Experimental Evaluation

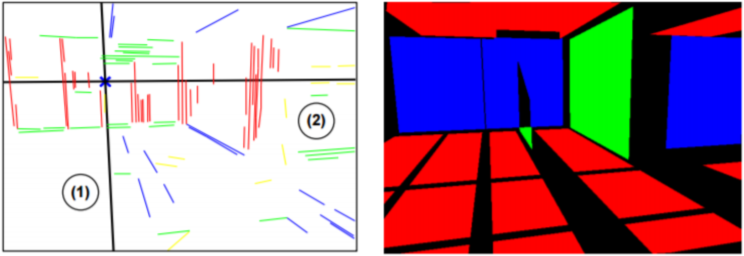
- RESULTS:

Geometric context from a single image: ground (green), sky (blue), vertical regions (red) subdivided into planar orientations (arrows) and non-planar solid ('x') and porous ('o').

Figure 9:

Experimental Evaluation

- RESULTS:

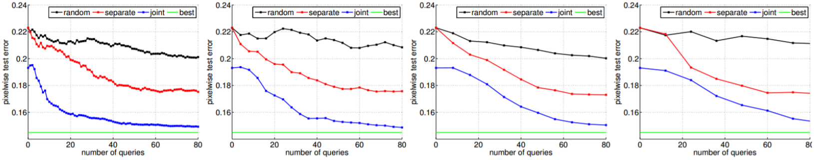


(a) (b)

Figure 10: Line segments and Orientation map. (a) Line segments, vanishing points, and vanishing lines. (b) Orientation map. Lines segments and regions are colored according to their orientation.

Experimental Evaluation

- RESULTS:

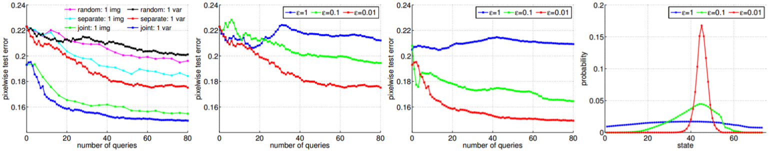


(a) $k = 1$ (b) $k = 4$ (c) $k = 8$ (d) $k = 12$

Figure 11: : Test set error as a function of the number of random variables labeled, when using joint vs separate active learning. The different plots reflect scenarios where the top k random variables are labeled at each iteration (i.e., batch setting). From left to right k = 1, 4, 8 and 12

Experimental Evaluation

- RESULTS:



(a) Image vs variable (b) ϵ separate (c) ϵ joint (d) Marginal distribution

Figure 12: Test set error as a function of the number of random variables labeled ((a)-(c)). Marginal distribution is illustrated in (d) for different ϵ

Experimental Evaluation

- RESULTS:

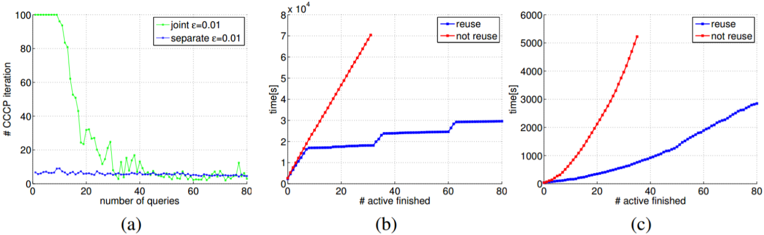


Figure 13: Number of CCCP iterations as a function of the amount of queried variables in (a) and time after specified number of active iterations in (b) (joint) and (c) (separate).

Overall Takeaways

Online Structured Prediction via Coactive Learning

- Defined a Coactive Learning Model, with a notion of linear utility and regret and algorithms that minimize it (preference perceptron)
- Extended to minimize any convex losses (convex preference perceptron).

Latent Structured Active Learning

- Reviewed latent structured prediction in a supervised setting.
- Designed active learning algorithms for latent structured prediction, using entropy as a decider for what subsets of the output space to label.