Preference Aggregation, Preference Elicitation, and Preference Learning

Paolo Viappiani <paolo.viappiani@lip6.fr>

Preferences

- Studied for long time from philosophers and logicians.
- Computer scientists are interested in computational tools for efficiently reasoning about preferences in order to provide personalized services.
- Applications: recommender systems, computational advertisement, intelligent user interfaces, cognitive assistants, personalized medecine, personal agents, robots,...

Preferences in AI Research

- Preference modelling
- Preference representation
- Preference reasoning
- Preference elicitation
- Preference learning

Preferences

Some of my interests:

- Preference (utility) elicitation
- Interactive social choice
- Ranking aggregation

Preferences

Some of my interests:

- Preference (utility) elicitation
- Interactive social choice
- Ranking aggregation

Some of the challenging questions:

- How to efficiently elicit user preferences (i.e. utilities) in an interactive way (by asking informative questions) in order to limit the cognitive burden posed to the user ?
  (utility elicitation)
- How do we elicit and aggregate the preferences of a group of users ?
  (interactive social choice)
- How do we learn preferences from a dataset of ranking data ?
  (ranking prediction and ranking clustering)

Recent AI trends in Incremental Elicitation

Key ideas:

- Optimal or near-optimal solutions even with partial preference information
- Incremental approach asking informative queries (a bit like active learning)

In which domains?

- Decision aid
- Recommender systems
- Social choice
- Sequential decision-making: MDPs and reinforcement learning
- Decision-making under uncertainty

Goal: Utility Maximization

Assume X is the set of possible choices (possibly very large)
The set X may be combinatorial, as the result of a constraint satisfaction problem, ...

The recommendation setting

- User's utility as weighted combination of item features

$$U(x) = w \cdot x$$

  where w and x are vectors.
- Recommendation as **utility maximization** problem subject to feasibility constratins:

$$x^* = \arg\max_{x \in X} w \cdot x$$

- Preference elicitation becomes the problem of learning utility weights w

Handling Preferences

We don't know *w*, but observe some user prefences

Handling Preferences

We don't know w, but observe some user prefences

"a is at least as preferred as b" is formalized by a constraint that weight vector w must satisfy

$$w \cdot (a - b) \geq 0$$

thus pairwise preferences induce linear constraints.

Handling Preferences

We don't know w, but observe some user prefences

"a is at least as preferred as b" is formalized by a constraint that weight vector w must satisfy

$$w \cdot (a - b) \geq 0$$

thus pairwise preferences induce linear constraints.

Handling noisy feedback

Users may not always state their true preferences!

- Allow constraints to violated by introduce a slack variable ε; the constraints representing preferences become: $w \cdot (a - b) \geq -\varepsilon$
- Several preferences, all with slack ε_i. We look for $w \in W$ such that, either all constraints are satisfied, or the sum of violations $\sum_i \varepsilon_i$ is small as possible.

Max-Margin Optimization

- Max-margin optimization aims at finding the weight w that explains the observed preferences as much as possible.
- Introduce a *shared* margin μ that all constraints have to satisfy
- Aim is to maximize μ while at the same time minimize the sum of the slacks

Optimization problem

$$\begin{aligned} \max \quad & \mu - \alpha||\varepsilon||_1 \\ s.t. \quad & w \cdot (a-b) \geq \mu - \varepsilon_{a,b} \ \ \forall (a,b) \in \mathcal{D} \\ & w_\perp \leq w_i \leq w_\top \\ & \varepsilon_{(a,b)} \geq 0 \ \ \forall (a,b) \in \mathcal{D} \end{aligned}$$

$\mathcal{D}$ is the set of binary preferences.

Setwise Max-Margin Optimization

Main idea

Learn a set of weight vectors, each representing a candidate utility function, maximizing diversity between the vectors while being as much as possible consistent with the available feedback.

Setwise Max-Margin Optimization

Main idea

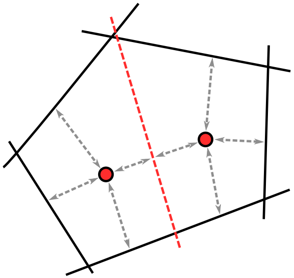
Learn a set of weight vectors, each representing a candidate utility function, maximizing diversity between the vectors while being as much as possible consistent with the available feedback.

- We now have k configuration vectors $x^1, \ldots, x^k$ and k weight vectors $w^1, \ldots, w^k$
- *User ranking constraints* for each of the weight vectors:

$$w^i \cdot (a - b) \geq \mu - \varepsilon^i_h$$

- *"Diversity" constraints* involving each pair w^i, w^j and x^i, x^j:

$$w^i \cdot (x^i - x^j) \geq \mu$$



Setwise Max-Margin Optimization

Setwise Optimization Problem

$$\max_{x,w,\varepsilon,\mu} \ \mu - \alpha \sum_{i=1}^{k} \|\varepsilon^i\|_1 - \beta \sum_{i=1}^{k} \|w^i\|_1 + \gamma \sum_{i=1}^{k} w^i \cdot x^i$$

$$\begin{aligned} \text{s.t.} \quad & w^i \cdot (a-b) \geq \mu - \varepsilon^i_{(a,b)} && \forall\, i \in [k], \forall\, (a,b) \in \mathcal{D} \\ & w^i \cdot (x^i - x^j) \geq \mu && \forall\, i,j \in [k], i \neq j \\ & w^\perp \leq w^i \leq w^\top && \forall\, i \in [k] \\ & x^i \in \mathcal{X}_{\text{feasible}} \ , \varepsilon^i \geq 0 && \forall\, i \in [k] \end{aligned}$$

The optimization retrieves both a set of *k* configurations and a set of *k* utility functions; these can be directly used for asking new queries.

[IJCAI 2016]

Reformulation

From quadratic formulation to mixed integer programming using some integer programming tricks.

MILP Forumulation

$$\max \quad \mu - \alpha \sum_{i=1}^{k} \|\varepsilon^i\|_1 - \beta \sum_{i=1}^{k} \|w^i\|_1 + \gamma \sum_{i=1}^{k} \sum_{z=1}^{m} p_z^{i,i}$$

$$\text{s.t.} \quad w^i(a-b) \geq \mu - \varepsilon^i_{a,b} \qquad \forall\, i \in [k], \forall\, (a,b) \in \mathcal{D}$$

$$\sum_{z=1}^{m} p_z^{i,i} - p_z^{i,j} \geq \mu \qquad \forall\, i,j \in [k], i \neq j$$

$$p_z^{i,i} \leq \min\{w_{\max} x_z^i,\; w_z^i\} \qquad \forall\, i,j \in [k], i \neq j, \forall\, z \in [m]$$

$$p_z^{i,j} \geq \max\{0,\; w_z^i - w_{\max}(1 - x_z^j)\} \qquad \forall\, i,j \in [k], i \neq j, \forall\, z \in [m]$$

$$w^{\perp} \leq w^i \leq w^{\top} \qquad \forall\, i \in [k]$$

$$x^i \in \mathcal{X}_{\text{feasible}}\;, \varepsilon^i \geq 0 \qquad \forall\, i \in [k]$$

The "Full" Elicitation Algorithm

```
procedure SetMargin($(k, \alpha, \beta, \gamma, T)$)
    $\mathcal{D} \leftarrow \emptyset$
    for $t = 1, \ldots, T$ do
        $\{\boldsymbol{w}^i, \boldsymbol{x}^i\}_{i=1}^k \leftarrow$ Solve($(\mathcal{D}, k, \alpha, \beta, \gamma)$)
        for $\boldsymbol{x}^i, \boldsymbol{x}^j \in \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k\}$ s.t. $i < j$ do
            $\mathcal{D} \leftarrow \mathcal{D} \cup$ QueryUser($(\boldsymbol{x}^i, \boldsymbol{x}^j)$)
        end for
    end for
    $\boldsymbol{w}^*, \boldsymbol{x}^* \leftarrow$ Solve($(\mathcal{D}, 1, \alpha, \beta, \gamma)$)
    return $\boldsymbol{w}^*, \boldsymbol{x}^*$
end procedure
```

The "Full" Elicitation Algorithm

```
procedure SetMargin($k, \alpha, \beta, \gamma, T$)
    $\mathcal{D} \leftarrow \emptyset$
    for $t = 1, \ldots, T$ do
        $\{\boldsymbol{w}^i, \boldsymbol{x}^i\}_{i=1}^k \leftarrow$ Solve($\mathcal{D}, k, \alpha, \beta, \gamma$)
        for $\boldsymbol{x}^i, \boldsymbol{x}^j \in \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^k\}$ s.t. $i < j$ do
            $\mathcal{D} \leftarrow \mathcal{D} \cup$ QueryUser($\boldsymbol{x}^i, \boldsymbol{x}^j$)
        end for
    end for
    $\boldsymbol{w}^*, \boldsymbol{x}^* \leftarrow$ Solve($\mathcal{D}, 1, \alpha, \beta, \gamma$)
    return $\boldsymbol{w}^*, \boldsymbol{x}^*$
end procedure
```

- Initially empty response set $\mathcal{D}$
- At each step solve the MILP
- Present the configurations $x^1, \ldots, x^k$ to the user by asking a series of pairwise queries
- New replies are added to $\mathcal{D}$.
- Final recommendation: optimization with set size $k = 1$

Experimental Setting

Simulated users

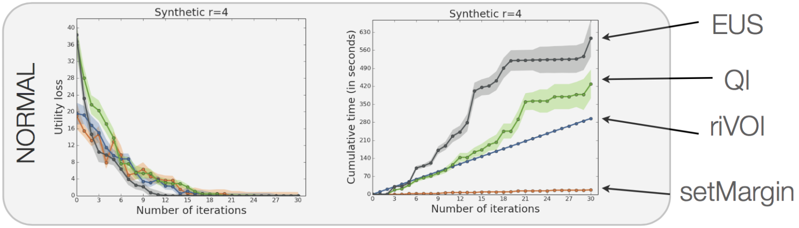
- Normal distribution
- Sparse normal distribution (80% of the weights set to zero)
- Indifference-augmented Bradley-Terry user response model
- Utility loss: quality of recommendation

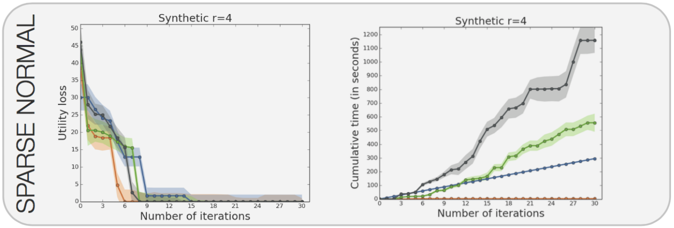
$$\max_{x \in X} u(x) - u(x^*)$$

  where x^* is the recommended object
- Settings:
  - Small synthetic configuration problems
  - Large “realistic” configuration scenarios
- Competitors: Bayesian methods

Experimental Results

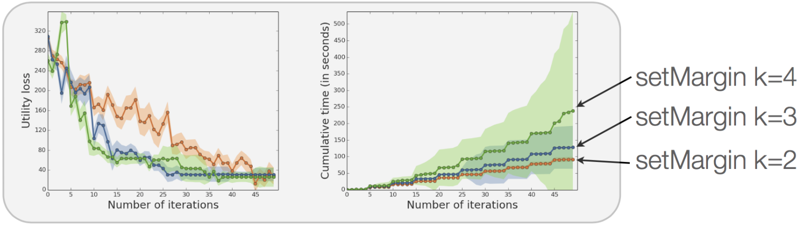
Synthetic dataset

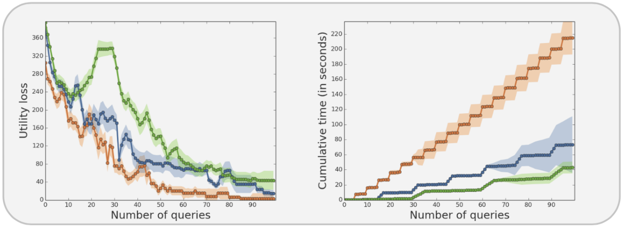




Experimental Results

Realistic dataset (PC recommendation task)





Multi-user setwise max-margin

Intuition

- Aim: to elicit simultaneously the preferences of M users
- Transfer preference knowledge from one user to other similar ones

Multi-user setwise max-margin

Intuition

- Aim: to elicit simultaneously the preferences of M users
- Transfer preference knowledge from one user to other similar ones

Each user is represented by a set of weights $w^{u,i}$; the *spread* of user u is:

$$v(u) := c \sum_{i \neq j} \|w^{u,i} - w^{u,j}\|^2$$

Multi-user setwise max-margin

Intuition

- Aim: to elicit simultaneously the preferences of M users
- Transfer preference knowledge from one user to other similar ones

Each user is represented by a set of weights $w^{u,i}$; the *spread* of user u is:

$$v(u) := c \sum_{i \neq j} \| w^{u,i} - w^{u,j} \|^2$$

Gaussian kernel for user “similarity”

$$k(u, y) := \exp \left(-\tau \sum_{i,j} \| w^{u,i} - w^{y,j} \|^2 \right)$$

Multi-user setwise max-margin

Intuition

- Aim: to elicit simultaneously the preferences of M users
- Transfer preference knowledge from one user to other similar ones

Each user is represented by a set of weights $w^{u,i}$; the *spread* of user u is:

$$v(u) := c \sum_{i \neq j} \| w^{u,i} - w^{u,j} \|^2$$

Gaussian kernel for user "similarity"

$$k(u,y) := \exp\left(-\tau \sum_{i,j} \| w^{u,i} - w^{y,j} \|^2 \right)$$

Aggregated estimated utility:

$$(1 - v(u))\, w^{u,i} \cdot x + v(u) \sum_{y \neq u} (1 - v(y)) k(u,y) w^{y} \cdot x$$

This is then plugged into the setwise maxmargin optimization.

Which user to ask?

- We should select the user in a “smart” way
  - Allow to significantly reduce residual uncertainty
  - Provide useful information about other “similar” users

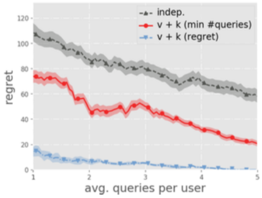
Which user to ask?

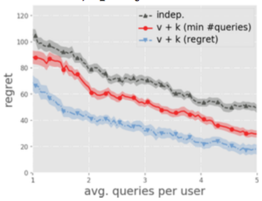
- We should select the user in a "smart" way
  - Allow to significantly reduce residual uncertainty
  - Provide useful information about other "similar" users

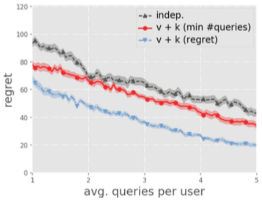
Experimental results (to appear in ADT 2017)

20 users, 1 cluster	20 users, 2 clusters	20 users, 5 clusters

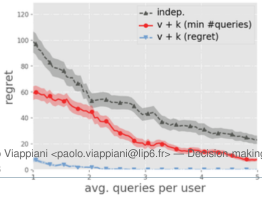
dense users, query set size $K = 2$

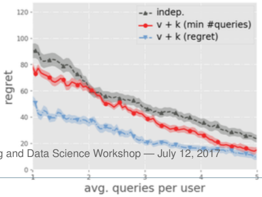


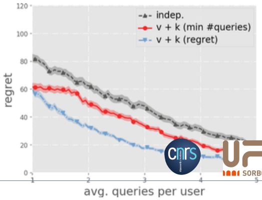




dense users, query set size $K = 3$







Preference Aggregation

Overview

- Preferences can be expressed in many forms
- Rankings are a popular format
- $\langle$Espresso, Capuccino, Tea, Americano$\rangle$.
- Ranking may be partial

What do we want to use rankings for?

- **Social choice:** take a decision together
- **Clustering:** summarize the preferences of a group of people

Computational Social Choice

Background

- Voting is an effective method for collective decision-making, used in political elections, technical committees, academic institutions.
- Interest in voting has increased in computer science both academically and in the industry
- Example applications: online web systems, group decision-making (scheduling a meeting), recommender systems

From standard social choice to incremental vote elicitation:

- Traditional voting schemes make stringent assumptions about the preference information provided by voters
- May be useful to deal with voting protocol requiring only partial preference information (e.g. partial rankings)
- Idea: *incremental elicitation* of preferences in a social choice context

Notation:

- set X of m alternatives
- n voters (or agents)
- Preference profile $v = (\sigma_1, \ldots, \sigma_n)$ where σ_i is a linear order (ranking)
- $\sigma(x)$ gives the rank of object x
- $s(a, v)$: score of item a in profile v
- A social choice function f maps a set of rankings (preference profiles) into a single winner:

$$f : v \to X$$

Some common social choice functions

- **Plurality**: the alternative with highest number of first position wins
- **Borda**: first position gives n points, second position $n-1, \ldots$
- **Scoring rules**: (generalizes both plurality and Borda)
  A score $w(r)$ is assigned to each position $r \in \{1, ..., n\}$.
  Each item is associated to an overall score $s(i) = \sum_{j=1}^{m} w(\sigma_j(i))$.
- **Maximin**
- **Copeland**
- $\ldots$

Social Choice with Partial Preference

Assume we are given a partial profile $p = (p_1, \ldots, p_n)$ (consisting of partial rankings).
Let $C(p)$ be the set of possible competitions of p (extending each element to a full ranking)

- **Necessary winner:** a choice that will be always be picked not matter how the preference rankings are extended

$$x \in NW \text{ iff } \forall C' \in C(p) \;\; x = f(C(p))$$

- **Possible winner:** a choice that may be a winner for at least a particular full extension of the preference rankings

$$x \in PW \text{ iff } \exists C' \in C(p) \quad : \quad x = f(C(p))$$

- *Example:* with plurality, if A has 10 votes, B has 9 and C has 7 and two users have not stated their most-preferred candidate

$$NW = \{\}; \;\; PW = \{A, B\}$$

- However, there will usually be too many possible winners and not enough necessary winners...

Minimax Regret

- Approximate winner determination with incomplete voter preferences
- Let $s(x, v)$ be the score obtained by x with profile v
- Definition of minimax regret

$$PMR(x, x', p) = \max_{v \in C(p)} s(x', v) - s(x, v)$$

$$MR(x, p) = \max_{a' \in A} PMR(x, x', p)$$

$$MMR(p) = \min_{a \in A} MR(x, p)$$

[Lu and Boutilier, 2011]

Observation

The regret-minimizing alternative may not be a possible winner for some voting rules.

Example

$$\begin{tikzcd}
 & a \arrow[d] & & & c \arrow[d] & & a \arrow[d] & & c \arrow[d] \\
 & b \arrow[dl] \arrow[d] \arrow[dr] & & & b \arrow[dl] \arrow[d] \arrow[dr] & & d \arrow[dr] & & e \arrow[dl] \\
c & d & e & a & d & e & & b &
\end{tikzcd}$$

k votes *k votes* *2k+1 votes*

Consider 2-approval, scoring rule with positional weights $(1, 1, 0, \dots, 0)$:

- Alternative *b* has exactly score $2k$
- One among *a* and *c* has a score of at least $2k + 1$
- *a* and *c* are possible winner (*b* is not)
- *b* has a regret of $k + 1$, *a* and *c* of $2k + 1$

Regret Computation

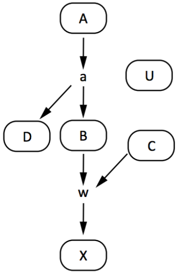
The regret can be decomposed over the users: [Lu and Boutilier, 2011]

$$\mathrm{PMR}(x, y, p) = \max_{v \in C(p)} s(y, v) - s(x, v) = \sum_i \max_{v_i \in C(p_i)} s(y, v_i) - s(x, v_i)$$

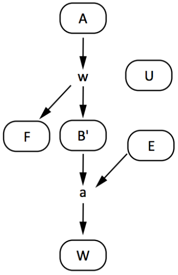
In practice we check the contribution to *max regret* for each individual user.

To compute contribution to regret PMR(a, w) of agent i:

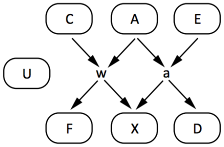
- Determine if we know whether the agent prefer a to w, prefer w to a or if they are currently incomparable
- Build the following lattices representing necessary preferences:



a > w



w > a



Inc(a,w)

For Borda, if $w \succ a$, the adversary will place all elements such that B is as large as possible. [Benabbou et al, 2016]: extension of this model to multi-attribute domains.

Ranking Aggregation and Clustering

Overview

- Preferences expressed as rankings; we consider aggregation with scoring rules.
  - Note: a full ranking is given as output !
- We study how scoring rules can be formulated as the minimization of some distance measures between rankings.
  - New family of aggregation methods, called *biased scoring rules*.
  - New distance functions, giving different weight to positions
  - We consider as well the case of partial rank data.
- Supervised learning: predict rank of missing items
- Unsupervised learning from ranking: clustering

Distance Measures for Rankings

- We are given a distance d on rankings and a set of input rankings $\{\sigma_1, ..., \sigma_m\}$
- Distances naturally lead to a way to generating an aggregate ranking:

$$\pi_d^* = \arg\min_{\pi \in S_n} \sum_{u=1}^{m} d(\pi, \sigma_u).$$

where S_n is the permutation group.
π_d^* is also called *median ranking* wrt d.

- Some common distances: Spearman, footrule, Kendall tau

Spearman

Spearman distance

Spearman distance is defined as taking the squares of the differences of the ranks:

$$d_S(\pi, \sigma) = \sum_{x \in X} [\pi(x) - \sigma(x)]^2.$$

Note that Spearman can be expressed as follows:
$d_S(\pi, \sigma) = \frac{n(n+1)(2n+1)}{3} - 2\sum_x \pi(x)\sigma(x)$.

Spearman

Spearman distance

Spearman distance is defined as taking the squares of the differences of the ranks:

$$d_S(\pi, \sigma) = \sum_{x \in X} [\pi(x) - \sigma(x)]^2.$$

Note that Spearman can be expressed as follows:
$d_S(\pi, \sigma) = \frac{n(n+1)(2n+1)}{3} - 2 \sum_x \pi(x)\sigma(x)$.

The connection between Borda and Spearman Distance

The Spearman distance *characterizes* the Borda rule:
$\pi^*_{Borda} = \arg\min_{\pi \in S_n} \sum_{u=1}^{m} d_S(\pi, \sigma_u)$.

Cfr. Theorem 2.2 in [John I Marden. Analyzing and modeling rank data. CRC Press, 1996].

- ⟨Espresso, Capuccino, *Tea*, *Americano*⟩ "very close" to ⟨Espresso, Capuccino, *Americano*, *Tea*⟩
- ⟨*Cappuccino*, *Espresso*, Tea, Americano⟩ not-so "close" to ⟨*Espresso*, *Capuccino*, Tea, Americano⟩

- ⟨Espresso, Capuccino, *Tea*, *Americano*⟩ "very close" to ⟨Espresso, Capuccino, *Americano*, *Tea*⟩
- ⟨*Cappuccino*, *Espresso*, Tea, Americano⟩ not-so "close" to ⟨*Espresso*, *Capuccino*, Tea, Americano⟩

Positional Spearman

We define Positional Spearman as a generalization of Spearman distance giving different weights to rank positions, computed as

$$d_{PS}(\pi, \sigma) = \sum_{x \in X} [w(\pi(x)) - w(\sigma(x))]^2 \quad (1)$$

parametrized by a vector w.

Characterization

Let w be strictly decreasing weights;the positional Spearman distance with weights w *characterizes* the scoring rule with the same weights:

$$\pi^*_{SR} = \arg\min_{\pi \in S_n} \sum_{u=1}^{m} d_{PS}(\pi, \sigma_u).$$

[IJCAI, 2015]

Biased Scoring Rules

- $s_{BSR}(x) = \phi_x + z_x \cdot s_{SR}(x)$
  parametrized by z_x (multiplicative bias) and ϕ_x (additive bonus).
- We characterize BSR with **item-weighting positional Spearman** d_{IPS}, allowing to weigh more the important items.

Biased Scoring Rules

- $s_{BSR}(x) = \phi_x + z_x \cdot s_{SR}(x)$
  parametrized by z_x (multiplicative bias) and ϕ_x (additive bonus).
- We characterize BSR with **item-weighting positional Spearman** d_{IPS}, allowing to weigh more the important items.

Incomplete Rank Data

- Aggregation of a set of *partial* input rankings: *expected Borda count*, *expected scoring rule*.
- Distances on partial rankings: $d_{E(PS)}(\pi, \sigma) = \mathbb{E}_{P(\hat{\sigma})}[d_s(\pi, \hat{\sigma})]$
  with $\hat{\sigma} \in S_n(\sigma)$ being a full ranking extending σ
- and similarly define $d_{E(PS)}$, $d_{E(IPS)}$.

Propositions

- *[Kamishima and Akaho, 2009]* $d_{E(S)}$ characterizes expected Borda count.
- $d_{E(PS)}$ characterizes the expected scoring rule.
- $d_{E(IPS)}$ characterizes the expected biased scoring rule.

Distance-based Clustering

$f : \{1,..,m\} \rightarrow \{1,..,k\}$ assignment of rankings to clusters.
The goal is to minimize the sum of distances:

$$(f^*, \bar{\pi}_1^*, .., \bar{\pi}_k^*) = \arg \min_{f, \bar{\pi}_1, \ldots, \bar{\pi}_k} \sum_{z=1}^{k} \sum_{j : f(j) = z} d(\bar{\pi}_z, \pi_j).$$

Algorithm: K-means adapted to rankings.

1. Randomly assign centroids
2. **Repeat** until convergence
    1. Assign rankings to closest centroid
    2. Recompute centroids
3. Return centroids, clustering

- *Centroid computation:* aggregate using a scoring rule; comp. time is linear in # input rankings and $O(n \log n)$ in # items.
- Distances d_{PS}, d_{IPS} can account for specific desired behaviors (more weight to top positions, bias in favor of particular items).

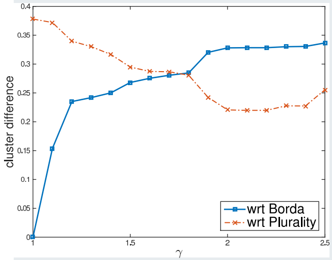
Computation time:

Running time is $O(nlogn)$ wrt # of items, and linear in m (# of rankings).
With 5000 rankings, clustering usually takes few seconds.

k	aggregator	distance	time	iterations
2	plurality	d_{PL}	0.21	3.00
	Borda	Spearman	6.99	10.10
	scoring rule	positional Spearman	4.71	6.05
	biased Borda	item-w Spearman	5.99	5.55
3	plurality	d_{PL}	0.30	3.00
	Borda	Spearman	13.26	13.15
	scoring rule	positional Spearman	8.58	7.05
	biased Borda	item-w Spearman	13.75	8.20
5	plurality	d_{PL}	0.47	3.00
	Borda	Spearman	19.92	11.70
	scoring rule	positional Spearman	16.08	7.95
	biased Borda	item-w Spearman	30.34	10.85
10	plurality	d_{PL}	0.98	3.00
	Borda	Spearman	49.17	14.15
	scoring rule	positional Spearman	37.55	9.50
	biased Borda	item-w Spearman	71.43	12.95

Sushi dataset (5000 rankings, 10 items).

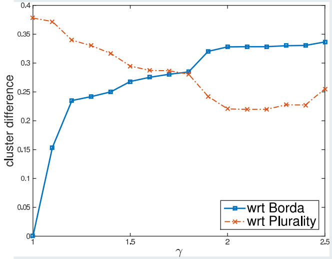
Experiment 1: Full Rank Data



Let w be such: $w_i - w_{i+1} = \gamma \cdot (w_{i+1} - w_{i+2})$ (γ controls steepness; $\gamma = 1 \rightarrow$ Borda).

- We measure the difference between clusterings (1−Rand index) wrt clustering with Borda and Plurality.
- When γ increases the clusters are increasingly similar to that obtained by Plurality ($\gamma \approx 1.8$ is a "middle ground").
- "Steeper" scoring rules tends to have larger concentration in a single cluster.

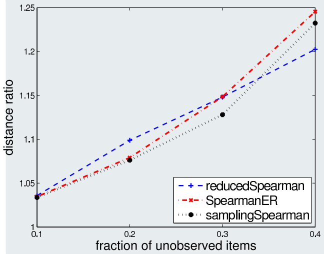
Experiment 1: Full Rank Data



Let w be such: $w_i - w_{i+1} = \gamma \cdot (w_{i+1} - w_{i+2})$ (γ controls steepness; $\gamma = 1 \rightarrow$ Borda).

- We measure the difference between clusterings (1−Rand index) wrt clustering with Borda and Plurality.
- When γ increases the clusters are increasingly similar to that obtained by Plurality ($\gamma \approx 1.8$ is a "middle ground").
- "Steeper" scoring rules tends to have larger concentration in a single cluster.

Experiment 2: Incomplete Rank Data



- Clustering using expected Borda and three distances on partial rankings (reduced Spearman, spearman with Expected Ranks, and sampling Spearman).
- The heuristic methods works almost as well as the more demanding approach based on sampling.

Conclusions

- Preferences are central to many AI systems
  - Active area of research (intersection of operation research, artificial intelligence, machine learning)
  - Incremental preference elicitation for one shot decisions, decision under risk, sequential decision-making, social choice
- A lot more going on:
  - *technical tools for dealing with complex criteria (e.g. handling monotonicity in Choquet), inference techniques for Bayesian learning of preferences, learning preference priors, eliciting rankings, sorting or outranking, Inverse RL, Bayesian RL with active feedback, lots of works in computational social choice, active collaborative filtering*

Thanks for your attention!