Active Learning to Rank using Pairwise Supervision

Buyue Qian Hongfei Li Jun Wang
Xiang Wang Ian Davidson

Learn to Rank

- Given a collection, specify an ordering of the items
- Approaches
    - Point-wise
    - Pair-wise
    - List-wise
- Applications
    - Information retrieval
    - Automated Essay Scoring
    - Parsing

Actively learn pair-wise ranking

- Query: $x_1 > x_2$?
- Three possible answers:
  - Strongly ordered
    - x1>x2
    - x1<x2
  - Weakly ordered
    - x1=x2

Task, formally...

- Learning objective: ranking function

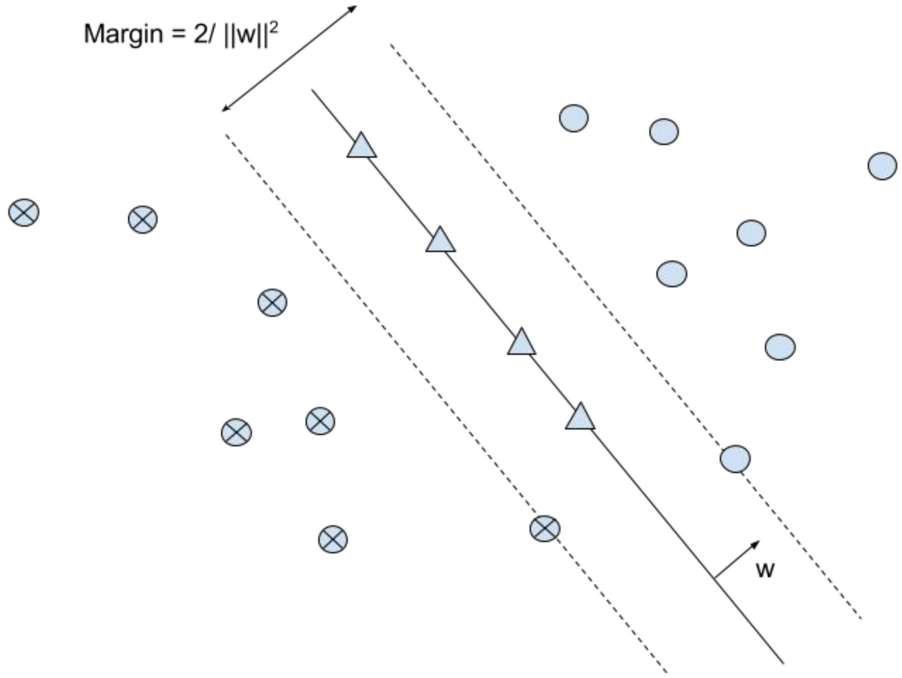
$$\tau(\mathbf{x}) = w^T \mathbf{x},$$

- Such that a maximum number of constraints are satisfied:

$$\begin{aligned} w^T \mathbf{x}_i &> w^T \mathbf{x}_j, \quad if \ (i,j) \in \mathcal{S} \\ w^T \mathbf{x}_i &= w^T \mathbf{x}_j, \quad if \ (i,j) \in \mathcal{W} \end{aligned}$$

- This is computationally intractable

As a large margin problem:



As an optimisation problem:

$$\min_{w,\xi,\gamma} \; \frac{1}{2} w^T w + C \sum_{i,j} (\xi_{ij} + \gamma_{ij})$$

$$\begin{aligned}
s.t. \qquad & w^T \mathbf{x}_i - w^T \mathbf{x}_j \geq 1 - \xi_{ij}, \quad if \; (i,j) \in \mathcal{S}; \\
& w^T \mathbf{x}_i - w^T \mathbf{x}_j \geq -\gamma_{ij}, \quad if \; (i,j) \in \mathcal{W}; \\
& \xi_{ij} \geq 0; \quad \gamma_{ij} \geq 0.
\end{aligned}$$

Kernalisation

- Problem: data vectors are most likely not linearly separable
- Map data points to higher dimensional space where they are linearly separable and learn

$$\tau(\mathbf{x}) = w^T \phi(\mathbf{x})$$

- Mapping could be very expensive, because the dimension of the space points are mapped to is very high
- Express the optimisation problem in terms of dot products of instance vectors, and use kernel function instead:

$$K(x, z) = \phi(x)^T \phi(z)$$

Kernelisation

$$\min_{w,\xi,\gamma} \max_{\alpha,\beta} \; L = \frac{1}{2} w^T w + C \sum_{i,j} (\xi_{ij} + \gamma_{ij})$$

$$- \sum_{(i,j)\in\mathcal{S}} \alpha_{ij} \left(w^T (\mathbf{x}_i - \mathbf{x}_j) - 1 + \xi_{ij} \right)$$

$$- \sum_{(i,j)\in\mathcal{W}} \beta_{ij} \left(w^T (\mathbf{x}_i - \mathbf{x}_j) + \gamma_{ij} \right)$$

$$s.t. \quad \xi_{ij} \geq 0; \quad \gamma_{ij} \geq 0.$$

$$\alpha ij \geq 0; \quad \beta_{ij} \geq 0.$$

Kernelisation

$$w = \sum_{(i,j)\in\mathcal{S}} \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) + \sum_{(i,j)\in\mathcal{W}} \beta_{ij}(\mathbf{x}_i - \mathbf{x}_j)$$

$$\tau_k(\mathbf{x}_t) = \sum_{(i,j)\in\mathcal{S}} \alpha_{ij}(K_{it} - K_{jt}) + \sum_{(i,j)\in\mathcal{W}} \beta_{ij}(K_{it} - K_{jt})$$

Kernelisation

$$\max_{\alpha,\beta} \sum_{(i,j)\in\mathcal{S}} \alpha_{ij}$$

$$-\frac{1}{2}\sum_{(i,j)\in\mathcal{S}}\sum_{(k,l)\in\mathcal{S}} \alpha_{ij}\alpha_{kl}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_k-\mathbf{x}_l)$$

$$-\frac{1}{2}\sum_{(i,j)\in\mathcal{W}}\sum_{(k,l)\in\mathcal{W}} \beta_{ij}\beta_{kl}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_k-\mathbf{x}_l)$$

$$-\sum_{(i,j)\in\mathcal{S}}\sum_{(k,l)\in\mathcal{W}} \alpha_{ij}\beta_{kl}(\mathbf{x}_i-\mathbf{x}_j)^T(\mathbf{x}_k-\mathbf{x}_l)$$

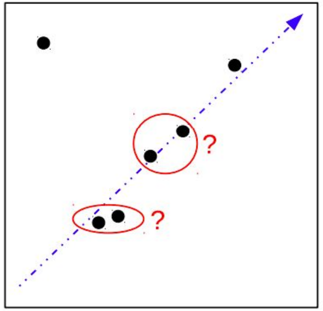
$$s.t. \qquad 0 \le \alpha_{ij} \le C; \quad 0 \le \beta_{ij} \le C.$$

Query strategy

- We want to choose a pair of items whose ordering would improve the ranking function
- Pair of elements whose ordering is “ambiguous”

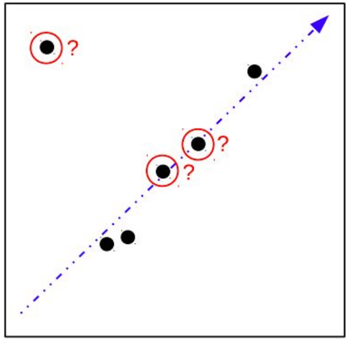
Local Uncertainty

- Ambiguous pair = the value that the ranking function returns is similar
- $$\mathcal{LU}(\mathbf{x}_i, \mathbf{x}_j) = |w^T(\mathbf{x}_i - \mathbf{x}_j)|^{-1}$$



- Their ranking being ambiguous might be desirable
  - They are similar instances and user would say that they are weakly ordered
  - This won't improve our hypothesis of ranking function
  - This information could be an important constraint later on when we know about some other pairs near these two points

Global Uncertainty



- Point whose ranking is ambiguous among others

$$\mathcal{D}(\mathbf{x}_i, \mathbf{x}_j) = \frac{w^T(\mathbf{x}_i - \mathbf{x}_j) - \min_{\mathbf{x}_k} w^T(\mathbf{x}_i - \mathbf{x}_k)}{\max_{\mathbf{x}_k} w^T(\mathbf{x}_i - \mathbf{x}_k) - \min_{\mathbf{x}_k} w^T(\mathbf{x}_i - \mathbf{x}_k)}$$

$$\mathcal{GU}(\mathbf{x}_i) = -\sum_{\mathbf{x}_j \in \mathbf{X}, j \neq i} \mathcal{D}(\mathbf{x}_i, \mathbf{x}_j) \log \mathcal{D}(\mathbf{x}_i, \mathbf{x}_j)$$

- Empirically points in a “dense” region get chosen
- If we only use this uncertainty measure, we might risk choosing outliers/noise which we don’t want our ranking function to fit around

Algorithm

- RankSVM
  - Gradient-based optimisation
  - Input data: strong ordering
  - No kernel
- Relative Attributes
  - Newton’s method
  - Input data: Strong + weak ordering
  - No kernel
- Proposed
  - RBF kernel

Comparison

- Different algorithm + Active learning
  - `RankSVM + Active`
  - `Relative Attributes + Active`
- Proposed method - global uncertainty
  - `Proposed + Local Uncertainty`
- Query about random pairs
  - `Proposed + Random`
  - `RankSVM + Random`
  - `Relative Attributes + Random`
- Proposed method

Evaluation metric

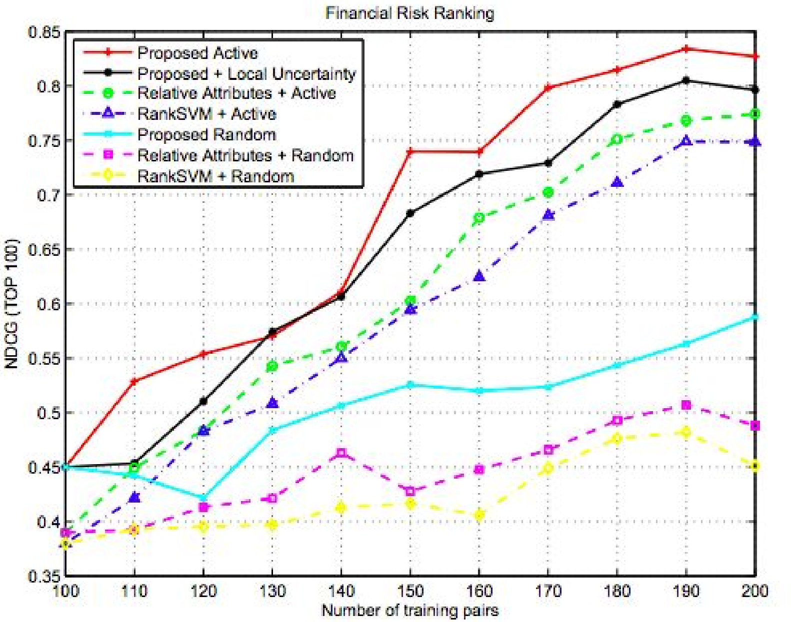
- Normalised discounted cumulative gain

$$\mathrm{NDCG}(\tau) = N \sum_{i} \frac{(2^{r(i)} - 1)}{\log_2(1 + i)}$$

Financial Risk Ranking

- Task: Rank companies w.r.t. their financial risks
- Feature:15k features from annual revenue reports of corporations projected to 100 dimensions
- Ground truth: stock return volatility measurements

Financial Risk Ranking

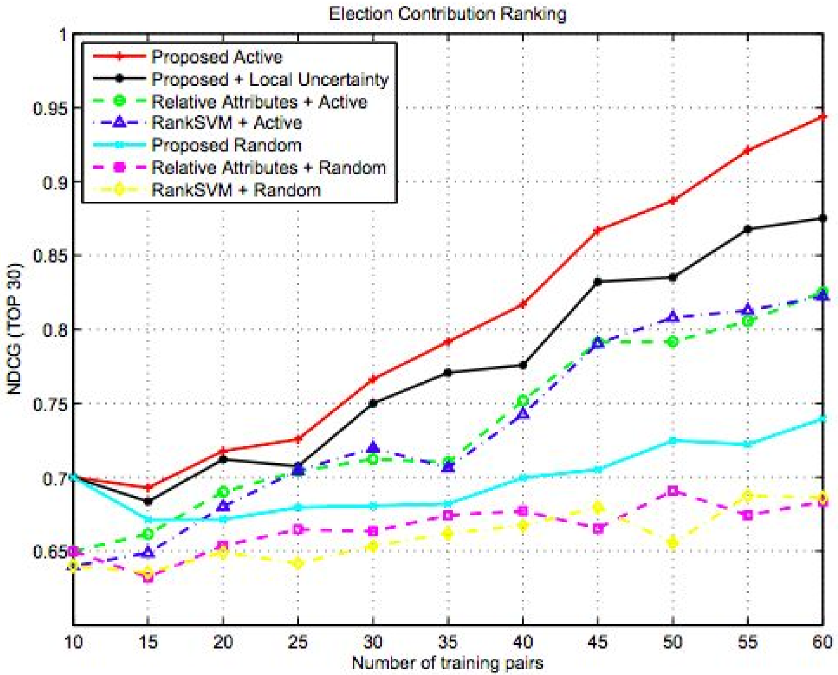


(a) Top 100

Election Votes Ranking

- Task: rank the 3,107 US counties w.r.t. their contributions in presidential election
- Feature: 6 dimension
    - Population > 18 yrs
    - Population with higher education
    - Number of owner-occupied housing units
    - Income
    - Latitude
    - Longitude
- Ground truth: log of the proportion of votes cast

Election Votes Ranking

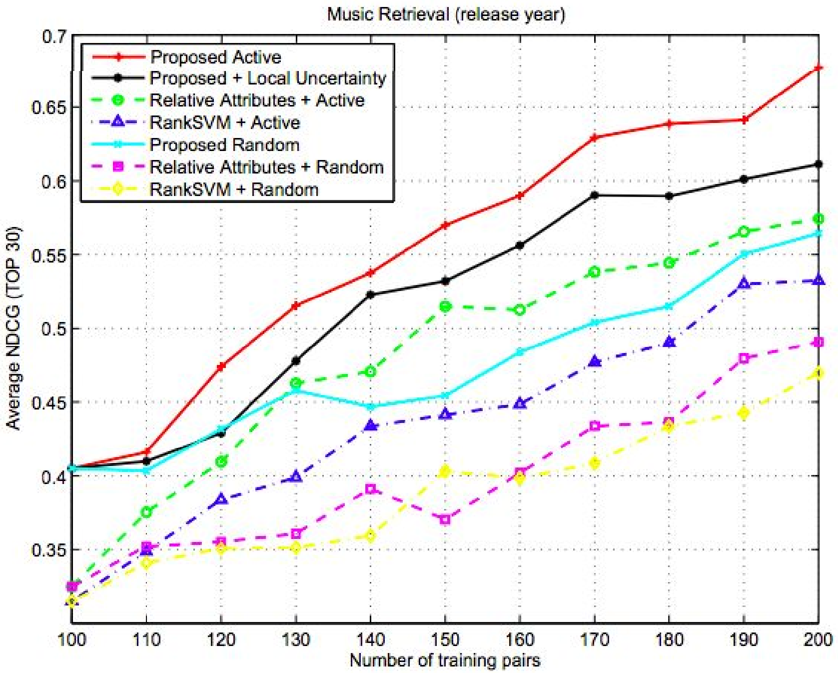


(a) Top 30

Musical Retrieval

- Task: to retrieve songs released in a particular year based on the features of audio content
- Feature: 90 dimension
- Weakly ordered if a music is produced ±3 year w.r.t. Music produced in the input year
- Strongly ordered below otherwise

Musical Retrieval



(a) Top 30

Discussion

- In general clear
- Improved performance attributed to both kernelisation and active learning
- Not clear what is “weakly ordered pair” in Application 1/2
- Comparison on execution speed
- In all cases it’s clearly not linearly separable (training pairs > feature)