Foundations of Machine Learning
Ranking

Mehryar Mohri
Courant Institute and Google Research
mohri@cims.nyu.edu
Motivation

- **Very large data sets:**
  - too large to display or process.
  - limited resources, need priorities.
  - ranking more desirable than classification.

- **Applications:**
  - search engines, information extraction.
  - decision making, auctions, fraud detection.

- **Can we learn to predict ranking accurately?**
Related Problem

- Rank aggregation: given $n$ candidates and $k$ voters each giving a ranking of the candidates, find ordering as close as possible to these.
  - Closeness measured in number of pairwise misrankings.
  - Problem NP-hard even for $k = 4$ (Dwork et al., 2001).
This Talk

- Score-based ranking
- Preference-based ranking
Score-Based Setting

- **Single stage**: learning algorithm
  - receives labeled sample of pairwise preferences;
  - returns scoring function $h: U \rightarrow \mathbb{R}$.

- **Drawbacks**:
  - $h$ induces a linear ordering for full set $U$.
  - does not match a query-based scenario.

- **Advantages**:
  - efficient algorithms.
  - good theory: VC bounds, margin bounds, stability bounds (FISS 03, RCMS 05, AN 05, AGHHR 05, CMR 07).
Score-Based Ranking

- **Training data**: sample of i.i.d. labeled pairs drawn from $U \times U$ according to some distribution $D$,

$$S = \left( (x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m) \right) \in U \times U \times \{-1, 0, +1\},$$

with $y_i = \begin{cases} 
+1 & \text{if } x'_i > \text{pref } x_i \\
0 & \text{if } x_i = \text{pref } x'_i \text{ or no information} \\
-1 & \text{if } x'_i < \text{pref } x_i.
\end{cases}$

- **Problem**: find hypothesis $h : U \rightarrow \mathbb{R}$ in $H$ with small generalization error

$$R(h) = \Pr_{(x, x') \sim D} \left[ (f(x, x') \neq 0) \land (f(x, x')(h(x') - h(x)) \leq 0) \right].$$
Empirical error:

\[
\hat{R}(h) = \frac{1}{m} \sum_{i=1}^{m} 1(y_i \neq 0) \land (y_i(h(x'_i) - h(x_i)) \leq 0) \cdot
\]

The relation \( x \mathcal{R} x' \iff f(x, x') = 1 \) may be non-transitive (needs not even be anti-symmetric).

Problem different from classification.
Distributional Assumptions

- Distribution over points: \( m \) points (literature).
  - labels for pairs.
  - squared number of examples \( O(m^2) \).
  - dependency issue.

- Distribution over pairs: \( m \) pairs.
  - label for each pair received.
  - independence assumption.
  - same (linear) number of examples.
Confidence Margin in Ranking

- Labels assumed to be in \{+1, -1\}.

- Empirical margin loss for ranking: for \( \rho > 0 \),

\[
\hat{R}_\rho(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho (y_i (h(x'_i) - h(x_i))) .
\]

\[
\hat{R}_\rho(h) \leq \frac{1}{m} \sum_{i=1}^{m} 1_{y_i [h(x'_i) - h(x_i)] \leq \rho} .
\]
Marginal Rademacher Complexities

- **Distributions:**
  - $D_1$ marginal distribution with respect to the first element of the pairs;
  - $D_2$ marginal distribution with respect to second element of the pairs.

- **Samples:**
  - $S_1 = ((x_1, y_1), \ldots, (x_m, y_m))$
  - $S_2 = ((x'_1, y_1), \ldots, (x'_m, y_m))$.

- **Marginal Rademacher complexities:**
  - $\mathcal{R}_{m}^{D_1}(H) = \mathbb{E}[\mathcal{R}_{S_1}(H)]$
  - $\mathcal{R}_{m}^{D_2}(H) = \mathbb{E}[\mathcal{R}_{S_2}(H)]$. 
Ranking Margin Bound

(Boyd, Cortes, MM, and Radovanovich 2012; MM, Rostamizadeh, and Talwalkar, 2012)

**Theorem**: let $H$ be a family of real-valued functions. Fix $\rho > 0$, then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample of size $m$, the following holds for all $h \in H$:

$$
R(h) \leq \hat{R}_\rho(h) + \frac{2}{\rho} \left( \mathcal{R}_{m1}^D(H) + \mathcal{R}_{m2}^D(H) \right) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.
$$
Ranking with SVMs

- **Optimization problem**: application of SVMs.

\[
\min_{\mathbf{w}, \xi} \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} \xi_i \\
\text{subject to: } y_i \left[ \mathbf{w} \cdot (\Phi(x'_i) - \Phi(x_i)) \right] \geq 1 - \xi_i \\
\xi_i \geq 0, \quad \forall i \in [1, m].
\]

- **Decision function**:

\[
h : x \mapsto \mathbf{w} \cdot \Phi(x) + b.
\]
The algorithm coincides with SVMs using feature mapping

\[(x, x') \mapsto \Psi(x, x') = \Phi(x') - \Phi(x).\]

Can be used with kernels:

\[K'(((x_i, x'_i), (x_j, x'_j)) = \Psi(x_i, x'_i) \cdot \Psi(x_j, x'_j)\]
\[= K(x_i, x_j) + K(x'_i, x'_j) - K(x'_i, x_j) - K(x_i, x'_j).\]

Algorithm directly based on margin bound.
Boosting for Ranking

- Use weak ranking algorithm and create stronger ranking algorithm.
- Ensemble method: combine base rankers returned by weak ranking algorithm.
- Finding simple relatively accurate base rankers often not hard.
- How should base rankers be combined?
CD RankBoost

(Freund et al., 2003; Rudin et al., 2005)

\[ H \subseteq \{0, 1\}^X, \epsilon_t^0 + \epsilon_t^+ + \epsilon_t^- = 1, \epsilon_t^s(h) = \Pr_{(x, x') \sim D_t} \left[ \text{sgn}(f(x, x')(h(x') - h(x))) = s \right]. \]

**RANKBOOST** (\(S = ((x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m))\))

1. for \(i \leftarrow 1 \text{ to } m \) do
2. \(D_1(x_i, x'_i) \leftarrow \frac{1}{m}\)
3. for \(t \leftarrow 1 \text{ to } T \) do
4. \(h_t \leftarrow \text{base ranker in } H \text{ with smallest } \epsilon_t^- - \epsilon_t^+ = -E_{i \sim D_t} \left[ y_i(h_t(x'_i) - h_t(x_i)) \right]\)
5. \(\alpha_t \leftarrow \frac{1}{2} \log \frac{\epsilon_t^+}{\epsilon_t^-}\)
6. \(Z_t \leftarrow \epsilon_t^0 + 2[\epsilon_t^+ \epsilon_t^-]^{1/2} \quad \triangleright \text{normalization factor}\)
7. for \(i \leftarrow 1 \text{ to } m \) do
8. \(D_{t+1}(x_i, x'_i) \leftarrow \frac{D_t(x_i, x'_i) \exp\left[-\alpha_t y_i(h_t(x'_i) - h_t(x_i))\right]}{Z_t}\)
9. \(\varphi_T \leftarrow \sum_{t=1}^T \alpha_t h_t\)
10. return \(\varphi_T\)
Notes

- Distributions $D_t$ over pairs of sample points:
  - originally uniform.
  - at each round, the weight of a misclassified example is increased.
  - observation: $D_{t+1}(x, x') = \frac{e^{-y[\varphi_t(x') - \varphi_t(x)]}}{|S| \prod_{s=1}^{t} Z_s}$, since

$$D_{t+1}(x, x') = \frac{D_t(x, x')e^{-y\alpha_t[h_t(x') - h_t(x)]}}{Z_t} = \frac{1}{|S|} \frac{e^{-y \sum_{s=1}^{t} \alpha_s[h_s(x') - h_s(x)]}}{\prod_{s=1}^{t} Z_s}.$$

- weight assigned to base classifier $h_t$: $\alpha_t$ directly depends on the accuracy of $h_t$ at round $t$. 
Coordinate Descent RankBoost

**Objective Function**: convex and differentiable.

\[
F(\alpha) = \sum_{(x,x',y) \in S} e^{-y[\varphi_T(x') - \varphi_T(x)]} = \sum_{(x,x',y) \in S} \exp \left( -y \sum_{t=1}^{T} \alpha_t [h_t(x') - h_t(x)] \right).
\]
• **Direction**: unit vector $e_t$ with

$$e_t = \arg\min_{t} \frac{dF(\alpha + \eta e_t)}{d\eta} \bigg|_{\eta=0} .$$

• **Since** $F(\alpha + \eta e_t) = \sum_{(x,x',y) \in S} e^{-y} \sum_{s=1}^{T} \alpha_s [h_s(x') - h_s(x)] e^{-\eta(y[h_t(x') - h_t(x)]},$

$$\frac{dF(\alpha + \eta e_t)}{d\eta} \bigg|_{\eta=0} = - \sum_{(x,x',y) \in S} y[h_t(x') - h_t(x)] \exp \left[ -y \sum_{s=1}^{T} \alpha_s [h_s(x') - h_s(x)] \right]$$

$$= - \sum_{(x,x',y) \in S} y[h_t(x') - h_t(x)] D_{T+1}(x,x') \left[ m \prod_{s=1}^{T} Z_s \right]$$

$$= - [\epsilon_t^+ - \epsilon_t^-] \left[ m \prod_{s=1}^{T} Z_s \right].$$

Thus, direction corresponding to base classifier selected by the algorithm.
• **Step size**: obtained via

\[
\frac{dF(\alpha + \eta e_t)}{d\eta} = 0
\]

\[
\Leftrightarrow - \sum_{(x,x',y) \in S} y[h_t(x') - h_t(x)] \exp \left[ - y \sum_{s=1}^{T} \alpha_s [h_s(x') - h_s(x)] \right] e^{-y[h_t(x')-h_t(x)]} \eta = 0
\]

\[
\Leftrightarrow - \sum_{(x,x',y) \in S} y[h_t(x') - h_t(x)] D_{T+1}(x, x') \left[ m \prod_{s=1}^{T} Z_s \right] e^{-y[h_t(x')-h_t(x)]} \eta = 0
\]

\[
\Leftrightarrow - \sum_{(x,x',y) \in S} y[h_t(x') - h_t(x)] D_{T+1}(x, x') e^{-y[h_t(x')-h_t(x)]} \eta = 0
\]

\[
\Leftrightarrow -[\epsilon_t^+ e^{-\eta} - \epsilon_t^- e^\eta] = 0
\]

\[
\Leftrightarrow \eta = \frac{1}{2} \log \frac{\epsilon_t^+}{\epsilon_t^-}.
\]

Thus, step size matches base classifier weight used in algorithm.
Bipartite Ranking

- **Training data:**
  - sample of negative points drawn according to $D_-$
    \[ S_- = (x_1, \ldots, x_m) \in U. \]
  - sample of positive points drawn according to $D_+$
    \[ S_+ = (x'_1, \ldots, x'_{m'}) \in U. \]

- **Problem:** find hypothesis $h : \mathcal{U} \rightarrow \mathbb{R}$ in $H$ with small generalization error
  \[
  R_D(h) = \Pr_{x \sim D_-, x' \sim D_+} \left[ h(x') < h(x) \right].
  \]
Connection between AdaBoost and RankBoost (Cortes & MM, 04; Rudin et al., 05).

- if constant base ranker used.
- relationship between objective functions.

More efficient algorithm in this special case (Freund et al., 2003).

Bipartite ranking results typically reported in terms of AUC.
AdaBoost and CD RankBoost

**Objective functions**: comparison.

\[
F_{\text{Ada}}(\alpha) = \sum_{x_i \in S_- \cup S_+} \exp (-y_i f(x_i)) \\
= \sum_{x_i \in S_-} \exp (+f(x_i)) + \sum_{x_i \in S_+} \exp (-f(x_i)) \\
= F_-(\alpha) + F_+(\alpha).
\]

\[
F_{\text{Rank}}(\alpha) = \sum_{(i,j) \in S_- \times S_+} \exp (-[f(x_j) - f(x_i)]) \\
= \sum_{(i,j) \in S_- \times S_+} \exp (+f(x_i)) \exp (-f(x_i)) \\
= F_-(\alpha)F_+(\alpha).
\]
AdaBoost and CD RankBoost

- **Property**: AdaBoost (non-separable case).
  - constant base learner \( h = 1 \) \( \rightarrow \) equal contribution of positive and negative points (in the limit).
  - consequence: AdaBoost asymptotically achieves optimum of CD RankBoost objective.

- **Observations**: if \( F_+(\alpha) = F_-(\alpha) \),

\[
d(F_{\text{Rank}}) = F_+ d(F_-) + F_- d(F_+) \\
= F_+ (d(F_-) + d(F_+)) \\
= F_+ d(F_{\text{Ada}}).
\]

(Rudin et al., 2005)
Bipartite RankBoost - Efficiency

Decomposition of distribution: for \((x, x') \in (S_-, S_+)\),
\[
D(x, x') = D_-(x)D_+(x').
\]

Thus,
\[
D_{t+1}(x, x') = \frac{D_t(x, x')e^{-\alpha_t[h_t(x') - h_t(x)]}}{Z_t}
= \frac{D_{t,-}(x)e^{\alpha_th_t(x)}}{Z_{t,-}} \cdot \frac{D_{t,+}(x')e^{-\alpha_th_t(x')}}{Z_{t,+}},
\]

with \(Z_{t,-} = \sum_{x \in S_-} D_{t,-}(x)e^{\alpha_th_t(x)}\) and \(Z_{t,+} = \sum_{x' \in S_+} D_{t,+}(x')e^{-\alpha_th_t(x')}\).
**ROC Curve**

Definition: the receiver operating characteristic (ROC) curve is a plot of the true positive rate (TP) vs. false positive rate (FP).

- TP: % positive points correctly labeled positive.
- FP: % negative points incorrectly labeled positive.

(Egan, 1975)
Area under the ROC Curve (AUC)

- **Definition**: the AUC is the area under the ROC curve. Measure of ranking quality.

- Equivalently,

\[
AUC(h) = \frac{1}{mm'} \sum_{i=1}^{m} \sum_{j=1}^{m'} 1_{h(x'_j) > h(x_i)} = \Pr_{x \sim \hat{D}_-} [h(x') > h(x)] \\
= 1 - \hat{R}(h).
\]
This Talk

- Score-based ranking
- Preference-based ranking
Preference-Based Setting

- **Definitions:**
  - $U$: universe, full set of objects.
  - $V$: finite query subset to rank, $V \subseteq U$.
  - $\tau^*$: target ranking for $V$ (random variable).

- **Two stages:** can be viewed as a reduction.
  - learn preference function $h: U \times U \rightarrow [0, 1]$.
  - given $V$, use $h$ to determine ranking $\sigma$ of $V$.

- **Running-time:** measured in terms of $|\text{calls to } h|$.
Preference-Based Ranking Problem

- **Training data**: pairs \((V, \tau^*)\) sampled i.i.d. according to \(D\):

\[
(V_1, \tau^*_1), (V_2, \tau^*_2), \ldots, (V_m, \tau^*_m) \quad V_i \subseteq U.
\]

subsets ranked by different labelers.

- **Problem**: for any query set \(V \subseteq U\), use \(h\) to return ranking \(\sigma_{h,V}\) close to target \(\tau^*\) with small average error

\[
R(h, \sigma) = \mathbb{E}_{(V, \tau^*) \sim D} [L(\sigma_{h,V}, \tau^*)].
\]
Preference Function

- $h(u, v)$ close to 1 when $u$ preferred to $v$, close to 0 otherwise. For the analysis, $h(u, v) \in \{0, 1\}$.

- Assumed pairwise consistent:
  \[ h(u, v) + h(v, u) = 1. \]

- May be non-transitive, e.g., we may have
  \[ h(u, v) = h(v, w) = h(w, u) = 1. \]

- Output of classifier or ‘black-box’.
Loss Functions

(for fixed \( (V, \tau^*) \))

- Preference loss:
  \[
  L(h, \tau^*) = \frac{2}{n(n - 1)} \sum_{u \neq v} h(u, v) \tau^*(v, u).
  \]

- Ranking loss:
  \[
  L(\sigma, \tau^*) = \frac{2}{n(n - 1)} \sum_{u \neq v} \sigma(u, v) \tau^*(v, u).
  \]
(Weak) Regret

- Preference regret:

$$
\mathcal{R}_{class}'(h) = \mathbb{E}_{V, \tau^*} [L(h|V, \tau^*)] - \mathbb{E}_{V} \min_{\tilde{h}} \mathbb{E}_{\tau^*|V} [L(\tilde{h}, \tau^*)].
$$

- Ranking regret:

$$
\mathcal{R}_{rank}'(A) = \mathbb{E}_{V, \tau^*, S} [L(A_s(V), \tau^*)] - \mathbb{E}_{V} \min_{\tilde{\sigma} \in S(V)} \mathbb{E}_{\tau^*|V} [L(\tilde{\sigma}, \tau^*)].
$$
Deterministic Algorithm

Stage one: standard classification. Learn preference function \( h : U \times U \rightarrow [0, 1] \).

Stage two: sort-by-degree using comparison function \( h \).
- sort by number of points ranked below.
- quadratic time complexity \( O(n^2) \).
Randomized Algorithm

(Ailon & MM, 08)

- **Stage one**: standard classification. Learn preference function \( h : U \times U \rightarrow [0, 1] \).

- **Stage two**: randomized QuickSort (Hoare, 61) using \( h \) as comparison function.

  - comparison function non-transitive unlike textbook description.
  
  - but, time complexity shown to be \( O(n \log n) \) in general.
Randomized QS

\[ h(v, u) = 1 \]

\[ h(u, v) = 1 \]

left recursion

right recursion

random pivot
Deterministic Algo. - Bipartite Case

\( V = V_+ \cup V_- \)  

- **Bounds:** for deterministic sort-by-degree algorithm
  
  - **expected loss:**
    \[
    \mathbb{E}_{V,\tau^*} [L(A(V), \tau^*)] \leq 2 \mathbb{E}_{V,\tau^*} [L(h, \tau^*)].
    \]
  
  - **regret:**
    \[
    \mathcal{R}'_{\text{rank}}(A(V)) \leq 2 \mathcal{R}'_{\text{class}}(h).
    \]
  
- **Time complexity:** \( \Omega(|V|^2) \).
Randomized Algo. - Bipartite Case

\( V = V_+ \cup V_- \)  

- **Bounds:** for randomized QuickSort (Hoare, 61).
  - expected loss (equality):
    
    \[
    E_{V, \tau^*, s} [L(Q^h_s(V), \tau^*)] = E_{V, \tau^*} [L(h, \tau^*)].
    \]
  - regret:
    \[
    \mathcal{R}'_{rank}(Q^h_s(\cdot)) \leq \mathcal{R}'_{class}(h).
    \]
- **Time complexity:**
  - full set: \( O(n \log n). \)
  - top \( k \): \( O(n + k \log k). \)
Proof Ideas

- **QuickSort decomposition:**

\[
p_{uv} + \frac{1}{3} \sum_{w \not\in \{u, v\}} p_{uvw} \left( h(u, w)h(w, v) + h(v, w)h(w, u) \right) = 1.
\]

- **Bipartite property:**

\[
\tau^*(u, v) + \tau^*(v, w) + \tau^*(w, u) = \\
\tau^*(v, u) + \tau^*(w, v) + \tau^*(u, w).
\]
**Lower Bound**

**Theorem:** for any deterministic algorithm $A$, there is a bipartite distribution for which

$$\mathcal{R}_{\text{rank}}(A) \geq 2 \mathcal{R}_{\text{class}}(h).$$

- thus, factor of $2$ is best in deterministic case.
- randomization necessary for better bound.

**Proof:** take simple case $U = V = \{u, v, w\}$ and assume that $h$ induces a cycle.
- up to symmetry, $A$ returns $u, v, w$ or $w, v, u$. 

\[ \begin{array}{c}
\text{u} \\
\text{v} \\
\text{h} \\
\text{w} \\
\end{array} \]
Lower Bound

- If $A$ returns $u, v, w$, then choose $\tau^*$ as:

- If $A$ returns $w, v, u$, then choose $\tau^*$ as:

\[
\begin{align*}
L[h, \tau^*] &= \frac{1}{3}; \\
L[A, \tau^*] &= \frac{2}{3}.
\end{align*}
\]
Guarantees - General Case

- Loss bound for QuickSort:

\[ E_{V,\tau^*,s}[L(Q_s^h(V), \tau^*)] \leq 2 E_{V,\tau^*}[L(h, \tau^*)]. \]

- Comparison with optimal ranking (see (CSS 99)):

\[
E_s[L(Q_s^h(V), \sigma_{optimal})] \leq 2 L(h, \sigma_{optimal}) \\
E_s[L(h, Q_s^h(V))] \leq 3 L(h, \sigma_{optimal}),
\]

where \( \sigma_{optimal} = \arg\min_{\sigma} L(h, \sigma). \)
Weight Function

- **Generalization:**

\[ \tau^*(u, v) = \sigma^*(u, v) \omega(\sigma^*(u), \sigma^*(v)). \]

- **Properties:** needed for all previous results to hold,
  - **symmetry:** \( \omega(i, j) = \omega(j, i) \) for all \( i, j \).
  - **monotonicity:** \( \omega(i, j), \omega(j, k) \leq \omega(i, k) \) for \( i < j < k \).
  - **triangle inequality:** \( \omega(i, j) \leq \omega(i, k) + \omega(k, j) \) for all triplets \( i, j, k \).
Weight Function - Examples

- **Kemeny:** \( w(i, j) = 1, \ \forall i, j. \)

- **Top-k:** \( w(i, j) = \begin{cases} 1 & \text{if } i \leq k \text{ or } j \leq k; \\ 0 & \text{otherwise.} \end{cases} \)

- **Bipartite:** \( w(i, j) = \begin{cases} 1 & \text{if } i \leq k \text{ and } j > k; \\ 0 & \text{otherwise.} \end{cases} \)

- **k-partite:** can be defined similarly.
(Strong) Regret Definitions

■ Ranking regret:

\[ R_{\text{rank}}(A) = \mathbb{E}_{V,\tau^*,S} [L(A_S(V), \tau^*)] - \min_{\tilde{\sigma}} \mathbb{E}_{V,\tau^*} [L(\tilde{\sigma}_V, \tau^*)]. \]

■ Preference regret:

\[ R_{\text{class}}(h) = \mathbb{E}_{V,\tau^*} [L(h|V, \tau^*)] - \min_{\tilde{h}} \mathbb{E}_{V,\tau^*} [L(\tilde{h}_V, \tau^*)]. \]

■ All previous regret results hold if for \( V_1, V_2 \supseteq \{u, v\}, \)

\[ \mathbb{E}_{\tau^*|V_1} [\tau^*(u, v)] = \mathbb{E}_{\tau^*|V_2} [\tau^*(u, v)] \]

for all \( u, v \) (pairwise independence on irrelevant alternatives).
References


References


References


