Table of Contents

1. Introduction

1.1 Motivation for Ranking

(1) Resource Limitation in Classification:

For large datasets, it's often impractical to process all items labeled as relevant by a classifier.

(2) Examples:

- Search engines
- Fraud detection systems

1.2 Two General Settings for Ranking

(1) Score-Based Setting: Margin-based generalization bounds using Rademacher complexity

(2) Preference-Based Setting: Regret-based guarantees for both deterministic and randomized algorithms.

1.3 Algorithms Discussed:

(1) SVM-Based Ranking Algorithm:

Derived from margin-based bounds

(2) RankBoost:

A boosting algorithm specifically for ranking

(3) Bipartite Ranking:

Classifying points into one of two classes and ranking

1.4 Evaluation Metrics

ROC Curves and AUC

2. The Problem of Ranking

2.1 Ranking Problem

Using labeled information to create an accurate ranking prediction for all points in a dataset.
In the score-based setting, labeled information is provided only for pairs of points and the quality of the predictor is measured by

its average pairwise misranking.

2.2 Scoring Function

Real-valued function assigning scores to input points

2.3 Preference Function f

(1) Definition

Let (X) be the input space and D be an unknown distribution over $X \times X$ indicating pairs of points the preference function $f : X \times X \to \{-1, 0, +1\}$.

(2) Non-Transitivity

- Example:
  $f(x, x') = 1$ and $f(x', x'') = 1$ but $f(x, x'') = -1$ can all hold simultaneously.
- Practice: This situation can arise in practice due to varying criteria for ranking different pairs of items.

2.4 Labeled Sample

The learner receives a labeled sample $S = \{((x_1, x_1'), y_1), \ldots, ((x_m, x_m'), y_m)\} \subset X \times X \times \{-1, 0, +1\}$ with (x_i, x_i') drawn i.i.d. according to D and $y_i = f(x_i, x_i')$.

2.5 Target

(1) Goal

To select a hypothesis $h \in H$ with small expected pairwise misranking (empirical error) or generalization error $R(h)$.

(2) Generalization Error $R(h)$

$$R(h) = \mathbb{P}_{(x,x') \sim \mathcal{D}} \left[\big(f(x, x') \neq 0 \big) \wedge \big(f(x, x')(h(x') - h(x)) \leq 0 \big) \right]$$

Measures the probability that h misranks a pair (x, x').

(3) Empirical Error $\hat{R}_S(h)$

$$\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1} \left[(y_i \neq 0) \wedge \big(y_i (h(x'_i) - h(x_i)) \leq 0 \big) \right]$$

Measures the misranking error of h over the sample S.

(4) Key points:

- The target preference function f is generally non-transitive whereas the scoring function h induces a transitive ordering.
- This inherent difference means no hypothesis h can perfectly predict the target pairwise ranking if f is not transitive.
- The empirical error is an estimate of the generalization error based on the sample data.

3. Generalization Bound

3.1 Margin-Based Generalization Bounds

(1) Simplification:

Pairwise labels are in $\{-1, +1\}$.

(2) Empirical Margin Loss

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

(3) Margin Loss and Pairwise Misranking

$$\hat{R}_{S,\rho}(h) \leq \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}_{y_i (h(x'_i) - h(x_i)) \leq \rho}$$

3.2 Theorem 10.1

(1) Distribution D

- D_1: Marginal distribution of the first element of the pairs in $X \times X$.
- D_2: Marginal distribution of the second element of the pairs.
- Rademacher Complexity:

$$\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) = \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})$$

If the D is symmetric.

(2) Margin Bound for Ranking

Theorem 10.1 (Margin bound for ranking) Let $\mathcal{H}$ be a set of real-valued functions. Fix $\rho > 0$; then, for any $\delta > 0$, with probability at least $1-\delta$ over the choice of a sample S of size m, each of the following holds for all $h \in \mathcal{H}$:

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\left(\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})\right) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \quad (10.5)$$

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\left(\hat{\mathfrak{R}}_{S_1}(\mathcal{H}) + \hat{\mathfrak{R}}_{S_2}(\mathcal{H})\right) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}} \quad (10.6)$$

(3) Proof:

- The proof is based on a comparison with the results of theorem 5.8.
- The Rademacher complexity bounds are used to derive the margin-based generalization bounds.

(4) Extension:

- Motivation: How these bounds can be generalized uniformly for all values of the margin parameter $\rho > 0$.

- Additional Term: $\sqrt{\frac{\log\log_2(2/\rho)}{m}}$,
  accounting for the complexity introduced by varying ρ, ensuring that the bound holds uniformly.
- Trade-off: Increasing ρ can make the middle term smaller; it makes the first term larger.
- Kernel-Based Hypotheses: Using known upper bounds for Rademacher complexity to derive explicit margin bounds for ranking.

3.3 Corollary 10.2

(1) Margin Bounds with Kernel-Based Hypotheses

Corollary 10.2 (Margin bounds for ranking with kernel-based hypotheses) Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel with $r = \sup_{x\in\mathcal{X}} K(x, x)$. Let $\Phi : \mathcal{X} \to \mathcal{H}$ be a feature mapping associated to K and let $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \Phi(x) : \|\mathbf{w}\|_{\mathcal{H}} \leq \Lambda\}$ for some $\Lambda \geq 0$. Fix $\rho > 0$. Then, for any $\delta > 0$, the following pairwise margin bound holds with probability at least $1 - \delta$ for any $h \in \mathcal{H}$:

$$R(h) \leq \hat{R}_{S,\rho}(h) + 4\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \quad (10.7)$$

(2) Implication:

- It depends only on the pairwise ranking margin and not directly on the dimension of the feature space.
- It suggests that a small generalization error can be achieved when ρ/r is large even if the empirical margin loss is small.

4. Ranking with SVMs

4.1 Theoretical Guarantee

Proceeding as in section 5.4 for classification, the guarantee of corollary 10.2 can be expressed as follows: for any $\delta > 0$, with probability at least $1 - \delta$, for all $h \in \mathcal{H} = \{x \mapsto \mathbf{w} \cdot \Phi(x) : \|\mathbf{w}\| \leq \Lambda\}$,

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m}\xi_i + 4\sqrt{\frac{r^2\Lambda^2}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}, \quad (10.8)$$

where $\xi_i = \max\left(1 - y_i\left[\mathbf{w} \cdot (\Phi(x_i') - \Phi(x_i))\right], 0\right)$ for all $i \in [m]$, and where $\Phi : \mathcal{X} \to \mathcal{H}$ is a feature mapping associated to a PDS kernel K.

4.2 Objective Function and Constraints

(1) Aim:

To minimize the right-hand side of the above inequality which consists of:

- Minimizing the sum of slack variables ξ_i.
- Minimizing $\|w\|^2$.

(2) Objective Function:

$$\min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i$$

(3) Subject to the Constraints:

subject to:
$y_i\left[\mathbf{w} \cdot (\Phi(x_i') - \Phi(x_i))\right] \geq 1 - \xi_i$
$\xi_i \geq 0, \quad \forall i \in [m].$

(4) Interpretation:

This problem is equivalent to the primal optimization problem of SVMs with a feature mapping Ψ: $\Psi(x, x') = \Phi(x') - \Phi(x)$

(5) Kernel Trick:

$$K_{ij}' = \Psi(x_i, x_i') \cdot \Psi(x_j, x_j') = K(x_i, x_j) + K(x_i', x_j') - K(x_i', x_j) - K(x_i, x_j') \quad (10.10)$$

for all $i, j \in [m]$.

By using a PDS kernel K, the equivalent dual problem can be expressed.

(6) Application:

- This algorithm can be applied to the ranking problem in the score-based setting.
- It can also be extended to cases where the labels are in $\{-1, 0, +1\}$.

5. RankBoost

5.1 Introduction:

Boosting algorithm for pairwise ranking like AdaBoost algorithm.

5.2 Components:

(1) Base Rankers:

Hypotheses returned by a weak learning algorithm for ranking.

(2) Weak Learning Algorithm:

An algorithm that generates base hypotheses with minimal accuracy.

5.3 Weighted error rate ϵ_t^s

(1) Meaning:

The weighted error rate of the base ranker at iteration t for pairs that fall into a specific category s.

(2) Definition:

$$\epsilon_t^s = \sum_{i=1}^{m} D_t(i) \mathbf{1}_{y_i(h_t(x_i') - h_t(x_i)) = s} = \mathbb{E}_{i \sim D_t} \left[\mathbf{1}_{y_i(h_t(x_i') - h_t(x_i)) = s} \right]$$

where $D_t(i)$ is the distribution weight of the i-th pair at iteration t.

5.3 Algorithm:

(1) Pseudocode

```
RankBoost($S = \{(x_1, x_1', y_1), \ldots, (x_m, x_m', y_m)\}$)

for i ← 1 to m do
    D_1(i) ← 1/m

for t ← 1 to T do
    h_t ← base ranker in H with smallest ε_t^- - ε_t^+
    α_t ← 1/2 log (ε_t^+/ε_t^-)
    Z_t ← ε_t^0 + 2(ε_t^+ ε_t^-)^1/2  // normalization factor

    for i ← 1 to m do
        D_{t+1}(i) ← D_t(i) exp[-α_t y_i (h_t(x_i') - h_t(x_i))] / Z_t

f ← Σ_{t=1}^T α_t h_t

return f
```

5.4 Key Components:

(1) Base Ranker Selection:

At each iteration, select h_t that minimizes $\epsilon_t^- - \epsilon_t^+$, focusing on the pair with the smallest pairwise misranking error:
$h_t \in \arg\min_{h \in \mathcal{H}} \{-\mathbb{E}_{i \sim D_t}[y_i (h(x_i') - h(x_i))]\}$

(2) Coefficient α_t:

A function of the ratio of misranking accuracies.

(3) Distribution Update:

The distribution D_t is updated to emphasize misranked pairs more in subsequent iterations.

5.5 Explanation:

(1) Updates:

If $\epsilon_t^- - \epsilon_t^+ > 0$, then $\frac{\epsilon_t^+}{\epsilon_t^-} < 1$ and $\alpha_t > 0$, meaning the new distribution D_{t+1} will increase the weight on misranked pairs and decrease the weight on correctly ranked pairs.

(2) Final Hypothesis:

After T rounds of boosting, the final hypothesis f is a linear combination of the base rankers:

$$f = \sum_{t=1}^{T} \alpha_t h_t.$$

(3) Distribution Update Identity:

The distribution $D_{t+1}(i)$ can be expressed in terms of the final predictor f_t and the normalization factors Z_s:

$$D_{t+1}(i) = \frac{e^{-y_i(f_t(x_i') - f_t(x_i))}}{m \prod_{s=1}^{t} Z_s}.$$

5.6 Bound on the Empirical Error:

(1) Theorem 10.3: Empirical Error Bound for RankBoost

Theorem 10.3 The empirical error of the hypothesis $h : \mathcal{X} \to \{0, 1\}$ returned by RankBoost verifies:

$$\hat{R}_S(h) \leq \exp\left[-2\sum_{t=1}^{T}\left(\frac{\epsilon_t^+ - \epsilon_t^-}{2}\right)^2\right]. \quad (10.13)$$

Furthermore, if there exists γ such that for all $t \in [T]$, $0 < \gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2}$, then

$$\hat{R}_S(h) \leq \exp(-2\gamma^2 T). \quad (10.14)$$

(2) Proof:

- General Inequality:

$$\mathbf{1}_{u \leq 0} \leq \exp(-u) \text{ for all } u \in \mathbb{R} \text{ and identity 10.12 :}$$

$$\hat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{1}_{y_i(f_T(x_i') - f_T(x_i)) \leq 0} \leq \frac{1}{m}\sum_{i=1}^{m} \exp\big(-y_i(f_T(x_i') - f_T(x_i))\big).$$

- Expression of Z_t:

$$\hat{R}_S(h) \leq \frac{1}{m}\sum_{i=1}^{m}\prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t.$$

- Definition and Upper Bound of Z_t:

$$Z_t = \epsilon_t^+ e^{-\alpha_t} + \epsilon_t^- e^{\alpha_t} + \epsilon_t^0$$

By using the simplification and convexity of the square root function:

$$Z_t \leq \exp\left(-\left(\frac{\epsilon_t^+ - \epsilon_t^-}{2}\right)^2\right).$$

- Assumption:

If there exists γ such that $0 < \gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2}$, then:

$$\prod_{t=1}^{T} Z_t \leq \exp(-2\gamma^2 T).$$

(3) Implications:

- Empirical Error:
  The empirical error decreases exponentially with the boosting rounds T.
- Edge Condition:
  The bound assumes that the difference of each base ranker is lower bounded by a positive value $\gamma > 0$.
- Weak Ranking Assumption:

The assumption $\gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2}$ can be relaxed to $\gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{\sqrt{\epsilon_t^+ + \epsilon_t^-}}$,

which considers the normalized relative difference.

- Coefficient α_t:
  It is chosen to minimize Z_t.
- Base Ranker Selection:
  Base ranker can be selected based on other criteria like minimal error on the distribution D_t.
- Range of Base Rankers:
  The base rankers could have a broader range.

5.7 Relationship with Coordinate Descent:

(1) Objective Function F:

$$F(\alpha) = \sum_{i=1}^{m} e^{-y_i [f_N(x_i') - f_N(x_i)]} = \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \alpha_j [h_j(x_i') - h_j(x_i)]},$$

A convex upper bound on the zero-one pairwise loss function.

(2) Parameter Update by Coordinate Descent:

- The parameter vector after iteration t is given by $\alpha_t = \alpha_{t-1} + \eta e_k$, where e_k is the unit vector corresponding to the k-th coordinate in $\mathbb{R}^N$.
- The function f_t is the linear combination of base hypotheses h_j up to iteration t:

$$f_t = \sum_{j=1}^{t} \alpha_j h_j.$$

(3) Distribution Update:

The distribution D_{t+1} over the indices $\{1, \ldots, m\}$ is updated as:

$$D_{t+1}(i) = \frac{e^{-y_i(f_t(x'_i) - f_t(x_i))}}{m \prod_{s=1}^{t} Z_s},$$

where Z_s is the normalization factor ensuring that D_{t+1} sums to 1.

(4) Directional Derivative and Coordinate Descent:

At each iteration $t \geq 1$, the direction e_k selected by coordinate descent minimizes the directional derivative $F'(\alpha_{t-1}, e_k)$:

$$F'(\alpha_{t-1}, e_k) = \lim_{\eta \to 0} \frac{F(\alpha_{t-1} + \eta e_k) - F(\alpha_{t-1})}{\eta}.$$

The directional derivative is expressed as:

$$F'(\alpha_{t-1}, e_k) = -\left[\epsilon_t^- - \epsilon_t^+\right] \prod_{s=1}^{t-1} Z_s.$$

(5) Step Size η:

The step size η is chosen to minimize $F(\alpha_{t-1} + \eta e_k)$. Setting the derivative with respect to η to zero:

$$\frac{d}{d\eta} F(\alpha_{t-1} + \eta e_k) = 0.$$

This results in:

$$\eta = \frac{1}{2}\log\frac{\epsilon_t^+}{\epsilon_t^-}.$$

(6) Alternative Loss Functions:

Other convex loss functions can be used to upper bound the zero-one pairwise misranking loss, such as the logistic loss:

$$\alpha \mapsto \sum_{i=1}^{m} \log(1 + e^{-y_i(f_N(x_i') - f_N(x_i))}),$$

which can lead to alternative boosting-type algorithms.

5.8 Margin Bound for Ensemble Methods in Ranking

(1) Assumptions:

- The pairwise labels are assumed to be in $\{-1, +1\}$.
- By lemma 7.4, the empirical Rademacher complexity of the convex hull $\mathrm{conv}(H)$ equals that of H.

(2) Corollary 10.4

Corollary 10.4 Let $\mathcal{H}$ be a set of real-valued functions. Fix $\rho > 0$; then, for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of a sample S of size m, each of the following ranking guarantees holds for all $h \in \mathrm{conv}(\mathcal{H})$:

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\big(\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})\big) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \quad (10.17)$$

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\Big(\hat{\mathfrak{R}}_{S_1}(\mathcal{H}) + \hat{\mathfrak{R}}_{S_2}(\mathcal{H})\Big) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \quad (10.18)$$

(3) Applications:

For RankBoost, these bounds apply to $f/\|\alpha\|_1$, where f is the hypothesis returned by the algorithm. Since f and $f/\|\alpha\|_1$ induce the same ordering of the points, for any $\delta > 0$, the following holds with probability at least $1-\delta$:

$$R(f) \leq \hat{R}_{S,\rho}(f/\|\alpha\|_1) + \frac{2}{\rho}\left(\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})\right) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \quad (10.19)$$

(4) Summary:

- Number of Boosting Rounds: The bound is independent of the number of boosting iterations.
- Dependencies of Bound:
  i. The margin ρ; ii. The sample size m; iii. The Rademacher complexity of the family of base classifiers H.
- Effective Generalization:
  The bounds ensure effective generalization if the pairwise margin loss is small for a sufficiently large margin ρ.

6. Bipartite ranking

6.1 Introduction

(1) Definition

- Bipartite ranking problem: In this scenario, the set of points is partitioned into two classes: the class of positive points & the class of negative points.
- Goal: To rank positive points higher than negative points.

(2) Traditional vs. Bipartite approach

- Traditional: The learner receives a sample of random pairs
- Bipartite ranking: The learner receives two separate samples

(3) Learning problem

- Generalization error

$$R(h) = \mathbb{P}_{x' \sim \mathcal{D}_+, x \sim \mathcal{D}_-}[h(x') < h(x)].$$

- Empirical error

$$\hat{R}_{S_+,S_-}(h) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{h(x_j) < h(x_i')}.$$

(4) Challenges

- Complexity: Requires dealing with mn pairs
- Differences from binary classification: Differs in objectives and measures of success

6.2 Boosting in bipartite ranking

(1) Key property of RankBoost

- Objective function: Bbased on the exponential function
- Distribution decomposition: The product of two distributions

(2) Efficiency in bipartite ranking

The time and space complexity depends only on the total number of points $m + n$, not on the number of pairs $m \times n$.

(3) Pseudocode

```
BipartiteRankBoost($S = \{x_1', \ldots, x_m', x_1, \ldots, x_n\}$)

for j ← 1 to m do
    $D_+^1(j) \leftarrow \frac{1}{m}$

for i ← 1 to n do
    $D_-^1(i) \leftarrow \frac{1}{n}$
```

```
for t ⟵ 1 to T do
    $h_t \leftarrow$ base ranker in $\mathcal{H}$ with smallest $\epsilon_t^- - \epsilon_t^+ = \mathbb{E}_{j \sim D_+^t}[h(x_j)] - \mathbb{E}_{i \sim D_-^t}[h(x_i)]$
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{\epsilon_t^+}{\epsilon_t^-}$
    $Z_t^+ \leftarrow 1 - \epsilon_t^+ + \sqrt{\epsilon_t^+ \epsilon_t^-}$

    for i ⟵ 1 to n do
        $D_+^{t+1}(i) \leftarrow D_+^t(i) \exp \left[ -\alpha_t h_t(x_i) \right] / Z_t^+$

    $Z_t^- \leftarrow 1 - \epsilon_t^- + \sqrt{\epsilon_t^+ \epsilon_t^-}$

    for j ⟵ 1 to m do
        $D_-^{t+1}(j) \leftarrow D_-^t(j) \exp \left[ +\alpha_t h_t(x_j) \right] / Z_t^-$

$f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$

return f
```

(4) Relationship with AdaBoost

- Objective function

The objective function of RankBoost can be expressed as:

$$F_{\text{RankBoost}}(\alpha) = \sum_{i=1}^{m} \sum_{j=1}^{n} \exp[-(f(x_i') - f(x_j))].$$

This can be decomposed into:

$$F_{\text{RankBoost}}(\alpha) = F_+(\alpha) F_-(\alpha),$$

where F_+ is the sum over positive points and F_- is the sum over negative points.

Similarly, the objective function of AdaBoost can be expressed as:

$$F_{\text{AdaBoost}}(\alpha) = F_+(\alpha) + F_-(\alpha).$$

- Connection

The gradient of the RankBoost objective function can be expressed in terms of the gradient of the AdaBoost objective function:

$$\nabla_\alpha F_{\text{RankBoost}}(\alpha) = F_-(\alpha)\nabla_\alpha F_+(\alpha) + F_+(\alpha)\nabla_\alpha F_-(\alpha).$$

If α minimizes F_{AdaBoost}, then $\nabla_\alpha F_{\text{RankBoost}}(\alpha) = 0$, implying α also minimizes the convex function associated with RankBoost.

- Empirical performance: It is observed empirically to be faster in convergence than AdaBoost.

6.3 Area under the ROC curve

(1) Definition

- AUC: Measures the performance of a ranking function by summarizing the trade-off between true positive rates and false positive rates across different thresholds.
- Pairwise misranking: The proportion of incorrectly ranked pairs $\hat{R}(h, U) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{1}_{h(z_i')<h(z_j)}$.
- AUC calculation: Representing the average pairwise ranking accuracy
- ROC curve: The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR)
- Construction of ROC curve: Points on the ROC curve are generated by varying a threshold value θ. At one extreme, all points are predicted as negatives, and at the other, all as positives.

(2) Properties of AUC

- Metrics: Higher AUC values indicate better ranking performance. An AUC of 1 means perfect ranking, while an AUC of 0.5 suggests random ranking.

- Computational complexity: i. Linear time calculation from a sorted array, given by $r/(mn)$; ii. The overall computational complexity is $O((m+n)\log(m+n))$ assuming a comparison-based sorting algorithm.

7. Preference-based setting

7.1 Introduction

(1) Comparison with score-based setting

- In the score-based setting, the goal is to provide a linear order for all points in X
- The preference-based setting simplifies the task by focusing only on ranking the query subset.

(2) Objective

The goal is to rank a finite query subset as accurately as possible

(3) Advantage

Unlike the score-based setting, the learning algorithm is not required to return a linear ordering of all points, which can be challenging due to non-transitive pairwise preference labeling.

(4) Stages

- First stage: learning the preference function

1. A sample of labeled pairs S is used to learn a preference function
2. The preference function can be obtained using a standard classification algorithm trained on S
3. h is not required to induce a linear ordering

- Second stage: ranking query subset

1. Given a query subset $X' \subseteq X$, the preference function h is used to determine a ranking of X'.

2. Generate an accurate ranking for the query subset.
3. The complexity of the algorithm that determines the ranking is measured by the number of calls to h.

7.2 Second-stage ranking problem

(1) Assumption

Pairwise consistent: $h(u, v) + h(v, u) = 1$ for all $u, v \in X$

(2) Stochastic characteristics

- Unknown Distribution D: Pairs (X, σ^*) are drawn from an unknown distribution, where $X \subseteq X$ is a query subset and σ^* is a target ranking or permutation of X.
- Objective: Use the preference function h to return an accurate ranking $A(X)$ that approximates σ^*.

(3) Loss function

- Loss function L

$$L(\sigma, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} \mathbf{1}_{\sigma(u)<\sigma(v)} \mathbf{1}_{\sigma^*(v)<\sigma^*(u)},$$

where the sum runs over all pairs (u, v) with distinct elements of X.

- Preference Function Loss $L(h, \sigma^*)$

$$L(h, \sigma^{)} = \frac{2}{n(n-1)} \sum_{u \neq v} h(u, v) \mathbf{1}_{\sigma^{(}v)<\sigma^*(u)}.$$

(4) Expected loss and regret

- Expected loss:

For a deterministic algorithm A, the expected loss is

$$\mathbb{E}_{(X,\sigma^*)\sim\mathcal{D}}[L(A(X),\sigma^*)].$$

- Regret of algorithm A:
  Defined as the difference between its loss and that of the best fixed global ranking:

$$\mathrm{Reg}(A) = \mathbb{E}_{(X,\sigma^*)\sim\mathcal{D}}[L(A(X),\sigma^*)] - \min_{\sigma'} \mathbb{E}_{(X,\sigma^*)\sim\mathcal{D}}[L(\sigma',\sigma^*)].$$

- Regret of preference function h:

$$\mathrm{Reg}(h) = \mathbb{E}_{(X,\sigma^*)\sim\mathcal{D}}[L(h|X,\sigma^*)] - \min_{h'} \mathbb{E}_{(X,\sigma^*)\sim\mathcal{D}}[L(h'|X,\sigma^*)],$$

where $h|X$ denotes the restriction of h to $X \times X$.

(5) Pairwise independence

- Assumes that for any $u, v \in X$, and any two sets X_1 and X_2 containing u and v, the random variable $\sigma^* \mid X$ conditioned on X maintains independence between pairs.
- Helps in defining and analyzing the regret for the ranking algorithms.

7.3 Deterministic algorithm

(1) Sort-by-degree algorithm

A deterministic approach that ranks elements based on how many other elements they are preferred to.

(2) Expected loss and regret

$$\mathbb{E}_{X,\sigma^*}[L(A_{\text{sort-by-degree}}(X),\sigma^*)] \leq 2\mathbb{E}_{X,\sigma^*}[L(h,\sigma^*)].$$

$$\mathrm{Reg}(A_{\text{sort-by-degree}}(X)) \leq 2\mathrm{Reg}(h).$$

(3) Implications

- The algorithm can achieve an accurate ranking when the loss or regret of the preference function is small.
- These bounds connect the ranking loss or regret of the algorithm to the classification loss or regret of h.

(4) Theorem 10.5

Theorem 10.5 (Lower bound for deterministic algorithms) For any deterministic algorithm A, there is a bipartite distribution for which

$$\mathrm{Reg}(A) \geq 2\mathrm{Reg}(h).$$

No deterministic algorithm can improve upon the factor of two in the regret guarantee

(5) Limitations

- Factor of two:
  The performance is at most twice as bad as the best possible.
- Example:
  For a binary classifier with an error rate of 25%, the worst-case pairwise misranking error for the ranking algorithm would be at most 50%.
- Randomization:
  Randomization is suggested as deterministic algorithms have an inherent lower bound on performance.

7.4 Randomized algorithm

(1)　Introduction

- General idea：　Extends the QuickSort algorithm for the second stage of ranking
- Advantage：　The expected time complexity is $O(n \log n)$ when applied to an array of size n, and it removes the factor of two in the deterministic case bounds.

(2) Algorithm

- Pivot selection:
  At each recursive step, a pivot element u is selected uniformly at random from X
- Partitioning:

For each $v \neq u$:

1. Place v on the left of u with probability $h(v, u)$.
2. Place v on the right of u with probability $h(u, v)$.

- Recursion: The algorithm proceeds recursively with the left and right subarrays.
- Concatenation:
  The final permutation is obtained by concatenating the results of the left recursion, u, and the right recursion.

(3) Guarantees

- Expected Loss

$$\mathbb{E}_{X,\sigma^*,s}[L(A_{\mathrm{QuickSort}}(X, s), \sigma^*)] = \mathbb{E}_{X,\sigma^*}[L(h, \sigma^*)].$$

- Regret

$$\mathrm{Reg}(A_{\mathrm{QuickSort}}) \leq \mathrm{Reg}(h).$$

- General Ranking Setting

$$\mathbb{E}_{X,\sigma^*,s}[L(A_{\mathrm{QuickSort}}(X, s), \sigma^*)] \leq 2\mathbb{E}_{X,\sigma^*}[L(h, \sigma^*)].$$

- Implication: The expected loss for QuickSort matches the loss of the preference function h without the factor of two

(4) Time complexity

Expected time complexity is $O(n \log n)$

7.5 Extension to other loss functions

(1) Weighted loss functions

The extension to L_ω allows for defining a family of loss functions that can emphasize different aspects of ranking, based on the weight function ω:

$$L_\omega(\sigma, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} \omega(\sigma^*(v), \sigma^*(u)) \mathbf{1}_{\sigma(u)<\sigma(v)} \mathbf{1}_{\sigma^*(v)<\sigma^*(u)},$$

where the sum runs over all pairs (u, v) with distinct elements of X.

(2) Weight function ω

It is a symmetric function, satisfying:

- Symmetry: $\omega(i, j) = \omega(j, i)$ for all i, j.
- Monotonicity: $\omega(i, j) \leq \omega(i, k)$ if either $i < j < k$ or $i > j > k$.
- Triangle Inequality: $\omega(i, j) \leq \omega(i, k) + \omega(k, j)$.

(3) Correct order importance

The triangle inequality property stems from the idea that if correctly ordering elements in positions (i, k) and (k, j) is not of great importance, then correctly ordering (i, j) should hold.

8. Other ranking criteria

8.1 Precision, precision@n, average precision, recall

(1) Precision

Measures the accuracy of positive predictions

(2) Precision @n

Focuses on the top n predictions

(3) Average precision

Averages precision at various cutoff points

(4) Recall

Measures the completeness of positive predictions

8.2 DCG and NDCG

(1) DCG

Uses relevance scores and discount factors to measure the quality of ranking

(2) NDCG

Normalizes DCG to allow comparison across different query sets