10 Ranking

The learning problem of ranking arises in many modern applications, including the design of search engines, information extraction platforms, and movie recommendation systems. In these applications, the ordering of the documents or movies returned is a critical aspect of the system. The main motivation for ranking over classification in the binary case is the limitation of resources: for very large data sets, it may be impractical or even impossible to display or process all items labeled as relevant by a classifier. A standard user of a search engine is not willing to consult all the documents returned in response to a query, but only the top ten or so. Similarly, a member of the fraud detection department of a credit card company cannot investigate thousands of transactions classified as potentially fraudulent, but only a few dozens of the most suspicious ones.

In this chapter, we study in depth the learning problem of ranking. We distinguish two general settings for this problem: the score-based and the preference-based settings. For the score-based setting, which is the most widely explored one, we present margin-based generalization bounds using the notion of Rademacher complexity. We then describe an SVM-based ranking algorithm that can be derived from these bounds and describe and analyze RankBoost, a boosting algorithm for ranking. We further study specifically the bipartite setting of the ranking problem where, as in binary classification, each point belongs to one of two classes. We discuss an efficient implementation of RankBoost in that setting and point out its connections with AdaBoost. We also introduce the notions of ROC curves and area under the ROC curve (AUC) which are directly relevant to bipartite ranking. For the preference-based setting, we present a series of results, in particular regret-based guarantees for both a deterministic and a randomized algorithm, as well as a lower bound in the deterministic case.

10.1 The problem of ranking

We first introduce the most commonly studied scenario of the ranking problem in machine learning. We will refer to this scenario as the *score-based setting* of the ranking problem. In section 10.6, we present and analyze an alternative setting, the *preference-based setting*.

The general supervised learning problem of ranking consists of using labeled information to define an accurate ranking prediction function for all points. In the scenario examined here, the labeled information is supplied only for pairs of points and the quality of a predictor is similarly measured in terms of its average pairwise misranking. The predictor is a real-valued function, a *scoring function*: the scores assigned to input points by this function determine their ranking.

Let $\mathcal{X}$ denote the input space. We denote by $\mathcal{D}$ an unknown distribution over $\mathcal{X} \times \mathcal{X}$ according to which pairs of points are drawn and by $f\colon \mathcal{X} \times \mathcal{X} \to \{-1, 0, +1\}$ a target labeling function or *preference function*. The three values assigned by f are interpreted as follows: $f(x, x') = +1$ if x' is preferred to x or ranked higher than x, $f(x, x') = -1$ if x is preferred to x', and $f(x, x') = 0$ if both x and x' have the same preference or ranking, or if there is no information about their respective ranking. This formulation corresponds to a deterministic scenario which we adopt for simplification. As discussed in section 2.4.1, it can be straightforwardly extended to a stochastic scenario where we have a distribution over $\mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$.

Note that in general no particular assumption is made about the transitivity of the order induced by f: we may have $f(x, x') = 1$ and $f(x', x'') = 1$ but $f(x, x'') = -1$ for three points x, x', and x''. While this may contradict an intuitive notion of preference, such preference orders are in fact commonly encountered in practice, in particular when they are based on human judgments. This is sometimes because the preference between two items are decided based on different features: for example, an individual may prefer movie x' to x because x' is an action movie and x a musical, and prefer x'' to x' because x'' is an action movie with more special effects than x'. Nevertheless, they may prefer x to x'' because the cost of the movie ticket for x'' is signicantly higher. Thus, in this example, two features, the genre and the price, are invoked, each affecting the decision for different pairs. In fact, in general, no assumption is made about the preference function, not even the antisymmetry of the order induced; thus, we may have $f(x, x') = 1$ and $f(x', x) = 1$ and yet $x \neq x'$.

The learner receives a labeled sample $S = \big((x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m)\big) \in \mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$ with $(x_1, x'_1), \ldots, (x_m, x'_m)$ drawn i.i.d. according to $\mathcal{D}$ and $y_i = f(x_i, x'_i)$ for all $i \in [m]$. Given a hypothesis set $\mathcal{H}$ of functions mapping $\mathcal{X}$ to $\mathbb{R}$, the ranking problem consists of selecting a hypothesis $h \in \mathcal{H}$ with small expected

pairwise misranking or generalization error $R(h)$ with respect to the target f:

$$R(h) = \mathop{\mathbb{P}}_{(x,x')\sim\mathcal{D}} \Big[\big(f(x,x') \neq 0\big) \wedge \big(f(x,x')(h(x') - h(x)) \leq 0\big) \Big]. \tag{10.1}$$

The empirical pairwise misranking or empirical error of h is denoted by $\widehat{R}_S(h)$ and defined by

$$\widehat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{(y_i \neq 0) \wedge (y_i(h(x_i') - h(x_i)) \leq 0)}\,. \tag{10.2}$$

Note that while the target preference function f is in general not transitive, the linear ordering induced by a scoring function $h \in \mathcal{H}$ is by definition transitive. This is a drawback of the score-based setting for the ranking problem since, regardless of the complexity of the hypothesis set $\mathcal{H}$, if the preference function is not transitive, no hypothesis $h \in \mathcal{H}$ can faultlessly predict the target pairwise ranking.

10.2 Generalization bound

In this section, we present margin-based generalization bounds for ranking. To simplify the presentation, we will assume for the results of this section that the pairwise labels are in $\{-1, +1\}$. Thus, if a pair (x, x') is drawn according to $\mathcal{D}$, then either x is preferred to x' or the opposite. The learning bounds for the general case have a very similar form but require more details. As in the case of classification, for any $\rho > 0$, we can define the empirical margin loss of a hypothesis h for pairwise ranking as

$$\widehat{R}_{S,\rho}(h) = \frac{1}{m} \sum_{i=1}^{m} \Phi_\rho(y_i(h(x_i') - h(x_i))), \tag{10.3}$$

where Φ_ρ is the margin loss function (definition 5.5). Thus, the empirical margin loss for ranking is upper bounded by the fraction of the pairs (x_i, x_i') that h is misranking or correctly ranking but with confidence less than ρ:

$$\widehat{R}_{S,\rho}(h) \leq \frac{1}{m} \sum_{i=1}^{m} 1_{y_i(h(x_i') - h(x_i)) \leq \rho}. \tag{10.4}$$

We denote by $\mathcal{D}_1$ the marginal distribution of the first element of the pairs in $\mathcal{X} \times \mathcal{X}$ derived from $\mathcal{D}$, and by $\mathcal{D}_2$ the marginal distribution with respect to the second element of the pairs. Similarly, S_1 is the sample derived from S by keeping only the first element of each pair: $S_1 = \big((x_1, y_1), \ldots, (x_m, y_m)\big)$ and S_2 the one obtained by keeping only the second element: $S_2 = \big((x_1', y_1), \ldots, (x_m', y_m)\big)$. We also denote by $\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H})$ the Rademacher complexity of $\mathcal{H}$ with respect to the marginal distribution $\mathcal{D}_1$, that is $\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) = \mathbb{E}[\widehat{\mathfrak{R}}_{S_1}(\mathcal{H})]$, and similarly $\mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H}) = \mathbb{E}[\widehat{\mathfrak{R}}_{S_2}(\mathcal{H})]$. Clearly,

if the distribution $\mathcal{D}$ is symmetric, the marginal distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ coincide and $\mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H}) = \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H})$.

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 $h \in \mathcal{H}$:*

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

$$R(h) \leq \widehat{R}_{S,\rho}(h) + \frac{2}{\rho}\big(\widehat{\mathfrak{R}}_{S_1}(\mathcal{H}) + \widehat{\mathfrak{R}}_{S_2}(\mathcal{H})\big) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}\,. \tag{10.6}$$

Proof: The proof is similar to that of theorem 5.8. Let $\widetilde{\mathcal{H}}$ be the family of hypotheses mapping $(\mathcal{X} \times \mathcal{X}) \times \{-1, +1\}$ to $\mathbb{R}$ defined by $\widetilde{\mathcal{H}} = \{z = ((x, x'), y) \mapsto y[h(x') - h(x)] \colon h \in \mathcal{H}\}$. Consider the family of functions $\widetilde{\mathcal{H}} = \{\Phi_\rho \circ f \colon f \in \widetilde{\mathcal{H}}\}$ derived from $\widetilde{\mathcal{H}}$ which are taking values in $[0, 1]$. By theorem 3.3, for any $\delta > 0$ with probability at least $1 - \delta$, for all $h \in \mathcal{H}$,

$$\mathbb{E}\left[\Phi_\rho(y[h(x') - h(x)])\right] \leq \widehat{R}_{S,\rho}(h) + 2\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}\,.$$

Since $1_{u \leq 0} \leq \Phi_\rho(u)$ for all $u \in \mathbb{R}$, the generalization error $R(h)$ is a lower bound on left-hand side, $R(h) = \mathbb{E}[1_{y[h(x')-h(x)]\leq 0}] \leq \mathbb{E}\left[\Phi_\rho(y[h(x') - h(x)])\right]$, and we can write:

$$R(h) \leq \widehat{R}_{S,\rho}(h) + 2\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}\,.$$

Since Φ_ρ is a $(1/\rho)$-Lipschitz function, by Talagrand's lemma (lemma 5.7) we have that $\mathfrak{R}_m\big(\Phi_\rho \circ \widetilde{\mathcal{H}}\big) \leq \frac{1}{\rho}\mathfrak{R}_m(\widetilde{\mathcal{H}})$. Here, $\mathfrak{R}_m(\widetilde{\mathcal{H}})$ can be upper bounded as follows:

$$\begin{aligned}
\mathfrak{R}_m(\widetilde{\mathcal{H}}) &= \frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\Big[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i y_i(h(x_i') - h(x_i))\Big] \\
&= \frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\Big[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i(h(x_i') - h(x_i))\Big] && (y_i\sigma_i \text{ and } \sigma_i\text{: same distrib.}) \\
&\leq \frac{1}{m}\mathop{\mathbb{E}}_{S,\sigma}\Big[\sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i') + \sup_{h\in\mathcal{H}}\sum_{i=1}^m \sigma_i h(x_i)\Big] && (\text{by sub-additivity of sup}) \\
&= \mathop{\mathbb{E}}_{S}\Big[\mathfrak{R}_{S_2}(\mathcal{H}) + \mathfrak{R}_{S_1}(\mathcal{H})\Big] && (\text{definition of } S_1 \text{ and } S_2) \\
&= \mathfrak{R}_m^{\mathcal{D}_2}(\mathcal{H}) + \mathfrak{R}_m^{\mathcal{D}_1}(\mathcal{H})\,,
\end{aligned}$$

which proves (10.5). The second inequality, (10.6), can be derived in the same way by using the second inequality of theorem 3.3, (3.4), instead of (3.3). □

These bounds can be generalized to hold uniformly for all $\rho > 0$ at the cost of an additional term $\sqrt{(\log \log_2(2/\rho))/m}$, as in theorem 5.9 and exercise 5.2. As for other margin bounds presented in previous sections, they show the conflict between two terms: the larger the desired pairwise ranking margin ρ, the smaller the middle term. However, the first term, the empirical pairwise ranking margin loss $\widehat{R}_{S,\rho}$, increases as a function of ρ.

Known upper bounds for the Rademacher complexity of a hypothesis set $\mathcal{H}$, including bounds in terms of VC-dimension, can be used directly to make theorem 10.1 more explicit. In particular, using theorem 10.1, we obtain immediately the following margin bound for pairwise ranking using kernel-based hypotheses.

Corollary 10.2 (Margin bounds for ranking with kernel-based hypotheses) *Let $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a PDS kernel with $r = \sup_{x \in \mathcal{X}} K(x,x)$. Let $\Phi\colon \mathcal{X} \to \mathbb{H}$ be a feature mapping associated to K and let $\mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x)\colon \|\mathbf{w}\|_{\mathbb{H}} \leq \Lambda\}$ 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 $h \in \mathcal{H}$:*

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

As with theorem 5.8, the bound of this corollary can be generalized to hold uniformly for all $\rho > 0$ at the cost of an additional term $\sqrt{(\log \log_2(2/\rho))/m}$. This generalization bound for kernel-based hypotheses is remarkable, since it does not depend directly on the dimension of the feature space, but only on the pairwise ranking margin. It suggests that a small generalization error can be achieved when ρ/r is large (small second term) while the empirical margin loss is relatively small (first term). The latter occurs when few points are either classified incorrectly or correctly but with margin less than ρ.

10.3 Ranking with SVMs

In this section, we discuss an algorithm that is derived directly from the theoretical guarantees just presented. The algorithm turns out to be a special instance of the SVM algorithm.

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 $h \in \mathcal{H} = \{x \mapsto \mathbf{w} \cdot \mathbf{\Phi}(x)\colon \|\mathbf{w}\| \leq \Lambda\}$,

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

where $\xi_i = \max\big(1 - y_i\big[\mathbf{w} \cdot \big(\mathbf{\Phi}(x_i') - \mathbf{\Phi}(x_i)\big)\big], 0\big)$ for all $i \in [m]$, and where $\mathbf{\Phi}\colon \mathcal{X} \to \mathbb{H}$ is a feature mapping associated to a PDS kernel K. An algorithm based on this

theoretical guarantee consists of minimizing the right-hand side of (10.8), that is minimizing an objective function with a term corresponding to the sum of the slack variables ξ_i, and another one minimizing $\|\mathbf{w}\|$ or equivalently $\|\mathbf{w}\|^2$. Its optimization problem can thus be formulated as

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi}} \ & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to: } & y_i\Big[\mathbf{w}\cdot\big(\boldsymbol{\Phi}(x_i') - \boldsymbol{\Phi}(x_i)\big)\Big] \geq 1 - \xi_i \\ & \xi_i \geq 0, \quad \forall i \in [m]\,. \end{aligned} \tag{10.9}$$

This coincides exactly with the primal optimization problem of SVMs, with a feature mapping $\boldsymbol{\Psi}\colon \mathcal{X}\times\mathcal{X}\to\mathbb{H}$ defined by $\boldsymbol{\Psi}(x,x') = \boldsymbol{\Phi}(x') - \boldsymbol{\Phi}(x)$ for all $(x,x')\in\mathcal{X}\times\mathcal{X}$, and with a hypothesis set of functions of the form $(x,x')\mapsto \mathbf{w}\cdot\boldsymbol{\Psi}(x,x')$. Thus, clearly, all the properties already presented for SVMs apply in this instance. In particular, the algorithm can benefit from the use of PDS kernels. Problem (10.9) admits an equivalent dual that can be expressed in terms of the kernel matrix $\mathbf{K}'$ defined by

$$\mathbf{K}'_{ij} = \boldsymbol{\Psi}(x_i,x_i')\cdot\boldsymbol{\Psi}(x_j,x_j') = K(x_i,x_j)+K(x_i',x_j')-K(x_i',x_j)-K(x_i,x_j'), \tag{10.10}$$

for all $i,j\in[m]$. This algorithm can provide an effective solution for pairwise ranking in practice. The algorithm can also be used and extended to the case where the labels are in $\{-1,0,+1\}$. The next section presents an alternative algorithm for ranking in the score-based setting.

10.4 RankBoost

This section presents a boosting algorithm for pairwise ranking, *RankBoost*, similar to the AdaBoost algorithm for binary classification. RankBoost is based on ideas analogous to those discussed for classification: it consists of combining different *base rankers* to create a more accurate predictor. The base rankers are hypotheses returned by a *weak learning algorithm* for ranking. As for classification, these base hypotheses must satisfy a minimal accuracy condition that will be described precisely later.

Let $\mathcal{H}$ denote the hypothesis set from which the base rankers are selected. Algorithm 10.1 gives the pseudocode of the RankBoost algorithm when $\mathcal{H}$ is a set of functions mapping from $\mathcal{X}$ to $\{0,1\}$. For any $s\in\{-1,0,+1\}$, we define ϵ_t^s by

$$\epsilon_t^s = \sum_{i=1}^{m}\mathcal{D}_t(i)1_{y_i(h_t(x_i')-h_t(x_i))=s} = \mathop{\mathbb{E}}_{i\sim\mathcal{D}_t}\big[1_{y_i(h_t(x_i')-h_t(x_i))=s}\big], \tag{10.11}$$

RankBoost$(S = ((x_1, x'_1, y_1) \ldots, (x_m, x'_m, y_m)))$

1 **for** $i \leftarrow 1$ **to** m **do**
2 $\quad \mathcal{D}_1(i) \leftarrow \frac{1}{m}$
3 **for** $t \leftarrow 1$ **to** T **do**
4 $\quad h_t \leftarrow$ base ranker in $\mathcal{H}$ with smallest $\epsilon_t^- - \epsilon_t^+ = - \mathop{\mathbb{E}}_{i \sim \mathcal{D}_t} \left[y_i \big(h_t(x'_i) - h_t(x_i) \big) \right]$
5 $\quad \alpha_t \leftarrow \frac{1}{2} \log \frac{\epsilon_t^+}{\epsilon_t^-}$
6 $\quad Z_t \leftarrow \epsilon_t^0 + 2[\epsilon_t^+ \epsilon_t^-]^{\frac{1}{2}}$ $\quad \triangleright$ normalization factor
7 $\quad$ **for** $i \leftarrow 1$ **to** m **do**
8 $\quad\quad \mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i) \exp \left[-\alpha_t y_i \big(h_t(x'_i) - h_t(x_i) \big) \right]}{Z_t}$
9 $f \leftarrow \sum_{t=1}^T \alpha_t h_t$
10 **return** f

Figure 10.1
RankBoost algorithm for $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$.

and simplify the notation ϵ_t^{+1} into ϵ_t^+ and similarly write ϵ_t^- instead of ϵ_t^{-1}. With these definitions, clearly the following equality holds: $\epsilon_t^0 + \epsilon_t^+ + \epsilon_t^- = 1$.

The algorithm takes as input a labeled sample $S = \big((x_1, x'_1, y_1), \ldots, (x_m, x'_m, y_m)\big)$ with elements in $\mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$, and maintains a distribution over the subset of the indices $i \in \{1, \ldots, m\}$ for which $y_i \neq 0$. To simplify the presentation, we will assume that $y_i \neq 0$ for all $i \in \{1, \ldots, m\}$ and consider distributions defined over $\{1, \ldots, m\}$. This can be guaranteed by simply first removing from the sample the pairs labeled with zero.

Initially (lines 1–2), the distribution is uniform ($\mathcal{D}_1$). At each round of boosting, that is at each iteration $t \in [T]$ of the loop 3–8, a new base ranker $h_t \in \mathcal{H}$ is selected with the smallest difference $\epsilon_t^- - \epsilon_t^+$, that is one with the smallest pairwise misranking error and largest correct pairwise ranking accuracy for the distribution $\mathcal{D}_t$:

$$h_t \in \mathop{\operatorname{argmin}}_{h \in \mathcal{H}} \Big\{ - \mathop{\mathbb{E}}_{i \sim \mathcal{D}_t} \Big[y_i \big(h(x'_i) - h(x_i) \big) \Big] \Big\}.$$

Note that $\epsilon_t^- - \epsilon_t^+ = \epsilon_t^- - (1 - \epsilon_t^- - \epsilon_t^0) = 2\epsilon_t^- + \epsilon_t^0 - 1$. Thus, finding the smallest difference $\epsilon_t^- - \epsilon_t^+$ is equivalent to seeking the smallest $2\epsilon_t^- + \epsilon_t^0$, which itself coincides with seeking the smallest ϵ_t^- when $\epsilon_t^0 = 0$. Z_t is simply a normalization factor to ensure that the weights $\mathcal{D}_{t+1}(i)$ sum to one. RankBoost relies on the assumption

that at each round $t \in [T]$, for the hypothesis h_t found, the inequality $\epsilon_t^+ - \epsilon_t^- > 0$ holds; thus, the probability mass of the pairs correctly ranked by h_t (ignoring pairs with label zero) is larger than that of misranked pairs. We denote by γ_t the *edge* of the base ranker h_t: $\gamma_t = \frac{\epsilon_t^+ - \epsilon_t^-}{2}$.

The precise reason for the definition of the coefficient α_t (line 5) will become clear later. For now, observe that if $\epsilon_t^+ - \epsilon_t^- > 0$, then $\epsilon_t^+ / \epsilon_t^- > 1$ and $\alpha_t > 0$. Thus, the new distribution $\mathcal{D}_{t+1}$ is defined from $\mathcal{D}_t$ by increasing the weight on i if the pair (x_i, x_i') is misranked ($y_i(h_t(x_i') - h_t(x_i) < 0$), and, on the contrary, decreasing it if (x_i, x_i') is ranked correctly ($y_i(h_t(x_i') - h_t(x_i) > 0$). The relative weight is unchanged for a pair with $h_t(x_i') - h_t(x_i) = 0$. This distribution update has the effect of focusing more on misranked points at the next round of boosting.

After T rounds of boosting, the hypothesis returned by RankBoost is f, which is a linear combination of the base classifiers h_t. The weight α_t assigned to h_t in that sum is a logarithmic function of the ratio of ϵ_t^+ and ϵ_t^-. Thus, more accurate base rankers are assigned a larger weight in that sum.

For any $t \in [T]$, we will denote by f_t the linear combination of the base rankers after t rounds of boosting: $f_t = \sum_{s=1}^t \alpha_t h_t$. In particular, we have $f_T = f$. The distribution $\mathcal{D}_{t+1}$ can be expressed in terms of f_t and the normalization factors Z_s, $s \in [t]$, as follows:

$$\forall i \in [m], \quad \mathcal{D}_{t+1}(i) = \frac{e^{-y_i(f_t(x_i')) - f_t(x_i)}}{m \prod_{s=1}^t Z_s}. \tag{10.12}$$

We will make use of this identity several times in the proofs of the following sections. It can be shown straightforwardly by repeatedly expanding the definition of the distribution over the point x_i:

$$\begin{aligned}
\mathcal{D}_{t+1}(i) &= \frac{\mathcal{D}_t(i) e^{-\alpha_t y_i (h_t(x_i') - h_t(x_i))}}{Z_t} \\
&= \frac{\mathcal{D}_{t-1}(i) e^{-\alpha_{t-1} y_i (h_{t-1}(x_i') - h_{t-1}(x_i))} e^{-\alpha_t y_i (h_t(x_i') - h_t(x_i))}}{Z_{t-1} Z_t} \\
&= \frac{e^{-y_i \sum_{s=1}^t \alpha_s (h_s(x_i') - h_s(x_i))}}{m \prod_{s=1}^t Z_s}.
\end{aligned}$$

10.4.1 Bound on the empirical error

We first show that the empirical error of RankBoost decreases exponentially fast as a function of the number of rounds of boosting when the edge γ_t of each base ranker h_t is lower bounded by some positive value $\gamma > 0$.

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

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

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

$$\widehat{R}_S(h) \leq \exp(-2\gamma^2 T)\,. \tag{10.14}$$

Proof: Using the general inequality $1_{u\leq 0} \leq \exp(-u)$ valid for all $u \in \mathbb{R}$ and identity 10.12, we can write:

$$\begin{aligned}\widehat{R}_S(h) = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i(f(x_i')-f(x_i))\leq 0} &\leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i(f(x_i')-f(x_i))} \\ &\leq \frac{1}{m}\sum_{i=1}^{m}\left[m\prod_{t=1}^{T} Z_t\right]\mathcal{D}_{T+1}(i) = \prod_{t=1}^{T} Z_t.\end{aligned}$$

By the definition of normalization factor, for all $t \in [T]$, we have $Z_t = \sum_{i=1}^{m} \mathcal{D}_t(i)$ $e^{-\alpha_t y_i(h_t(x_i')-h_t(x_i))}$. By grouping together the indices i for which $y_i(h_t(x_i')-h_t(x_i))$ takes the values in $+1$, -1, or 0, Z_t can be rewritten as

$$Z_t = \epsilon_t^+ e^{-\alpha_t} + \epsilon_t^- e^{\alpha_t} + \epsilon_t^0 = \epsilon_t^+\sqrt{\frac{\epsilon_t^-}{\epsilon_t^+}} + \epsilon_t^-\sqrt{\frac{\epsilon_t^+}{\epsilon_t^-}} + \epsilon_t^0 = 2\sqrt{\epsilon_t^+\epsilon_t^-} + \epsilon_t^0\,.$$

Since $\epsilon_t^+ = 1 - \epsilon_t^- - \epsilon_t^0$, we have

$$4\epsilon_t^+\epsilon_t^- = (\epsilon_t^+ + \epsilon_t^-)^2 - (\epsilon_t^+ - \epsilon_t^-)^2 = (1-\epsilon_t^0)^2 - (\epsilon_t^+ - \epsilon_t^-)^2.$$

Thus, assuming that $\epsilon_t^0 < 1$, Z_t can be upper bounded as follows:

$$\begin{aligned}Z_t &= \sqrt{(1-\epsilon_t^0)^2 - (\epsilon_t^+ - \epsilon_t^-)^2} + \epsilon_t^0 \\ &= (1-\epsilon_t^0)\sqrt{1 - \frac{(\epsilon_t^+ - \epsilon_t^-)^2}{(1-\epsilon_t^0)^2}} + \epsilon_t^0 \\ &\leq \sqrt{1 - \frac{(\epsilon_t^+ - \epsilon_t^-)^2}{(1-\epsilon_t^0)}} \\ &\leq \exp\left(-\frac{(\epsilon_t^+ - \epsilon_t^-)^2}{2(1-\epsilon_t^0)}\right) \leq \exp\left(-\frac{(\epsilon_t^+ - \epsilon_t^-)^2}{2}\right) = \exp\left(-2[(\epsilon_t^+ - \epsilon_t^-)/2]^2\right),\end{aligned}$$

where we used for the first inequality the concavity of the square-root function and the fact that $0 < 1 - \epsilon_t^0 \leq 1$, and the inequality $1 - x \leq e^{-x}$ valid for all $x \in \mathbb{R}$ for the second inequality. This upper bound on Z_t also trivially holds when $\epsilon_t^0 = 1$ since in that case $\epsilon_t^+ = \epsilon_t^- = 0$. This concludes the proof. □

As can be seen from the proof of the theorem, the weak ranking assumption $\gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2}$ with $\gamma > 0$ can be replaced with the somewhat weaker requirement $\gamma \leq \frac{\epsilon_t^+ - \epsilon_t^-}{2\sqrt{1-\epsilon_t^0}}$, with $\epsilon_t^0 \neq 1$, which can be rewritten as $\gamma \leq \frac{1}{2}\frac{\epsilon_t^+ - \epsilon_t^-}{\sqrt{\epsilon_t^+ + \epsilon_t^-}}$, with $\epsilon_t^+ + \epsilon_t^- \neq 0$, where the quantity $\frac{\epsilon_t^+ - \epsilon_t^-}{\sqrt{\epsilon_t^+ + \epsilon_t^-}}$ can be interpreted as a (normalized) relative difference between ϵ_t^+ and ϵ_t^-.

The proof of the theorem also shows that the coefficient α_t is selected to minimize Z_t. Thus, overall, these coefficients are chosen to minimize the upper bound on the empirical error $\prod_{t=1}^T Z_t$, as for AdaBoost. The RankBoost algorithm can be generalized in several ways:

- Instead of a hypothesis with minimal difference $\epsilon_t^- - \epsilon_t^+$, h_t can be more generally a base ranker returned by a weak ranking algorithm trained on $\mathcal{D}_t$ with $\epsilon_t^+ > \epsilon_t^-$;
- The range of the base rankers could be $[0, +1]$, or more generally $\mathbb{R}$. The coefficients α_t can then be different and may not even admit a closed form. However, in general, they are chosen to minimize the upper bound $\prod_{t=1}^T Z_t$ on the empirical error.

10.4.2 Relationship with coordinate descent

RankBoost coincides with the application of the coordinate descent technique to a convex and differentiable objective function F defined for all samples $S = \big((x_1, x_1', y_1), \ldots, (x_m, x_m', y_m)\big) \in \mathcal{X} \times \mathcal{X} \times \{-1, 0, +1\}$ and $\bar{\boldsymbol{\alpha}} = (\bar{\alpha}_1, \ldots, \bar{\alpha}_n) \in \mathbb{R}^N$, $N \geq 1$ by

$$F(\bar{\boldsymbol{\alpha}}) = \sum_{i=1}^m e^{-y_i[f_N(x_i') - f_N(x_i)]} = \sum_{i=1}^m e^{-y_i \sum_{j=1}^N \bar{\alpha}_j [h_j(x_i') - h_j(x_i)]}, \tag{10.15}$$

where $f_N = \sum_{j=1}^N \bar{\alpha}_j h_j$. This loss function is a convex upper bound on the zero-one pairwise loss function $\bar{\boldsymbol{\alpha}} \mapsto \sum_{i=1}^m 1_{y_i[f_N(x_i') - f_N(x_i)] \leq 0}$, which is not convex. Let $\mathbf{e}_k$ denote the unit vector corresponding to the kth coordinate in $\mathbb{R}^N$ and let $\bar{\boldsymbol{\alpha}}_{t-1} = \bar{\boldsymbol{\alpha}}_t + \eta \mathbf{e}_k$ denote the parameter vector after iteration t (with $\bar{\boldsymbol{\alpha}}_0 = \mathbf{0}$). Also, for any $t \in [T]$, we define the function $\bar{f}_t = \sum_{j=1}^N \bar{\alpha}_{t,j} h_j$ and distribution $\bar{\mathcal{D}}_t$ over the indices $\{1, \ldots, m\}$ as follows:

$$\bar{\mathcal{D}}_{t+1}(i) = \frac{e^{-y_i(\bar{f}_t(x_i')) - \bar{f}_t(x_i))}}{m \prod_{s=1}^t \bar{Z}_s}, \tag{10.16}$$

where $\bar{Z}_t$ is the analogue of Z_t computed as a function of $\bar{\mathcal{D}}_t$ instead of $\mathcal{D}_t$. Similarly, let $\bar{\epsilon}_t^+, \bar{\epsilon}_t^-$, and $\bar{\epsilon}_t$ denote the analogues of $\epsilon_t^+, \epsilon_t^-$, and ϵ_t defined with respect to $\bar{\mathcal{D}}_t$ instead of $\mathcal{D}_t$.

Following very similar steps as in section 7.2.2, we will use an inductive argument argument to show that coordinate descent on F and the RankBoost algorithm are

in fact equivalent. Clearly, $\bar{\mathcal{D}}_{t+1} = \mathcal{D}_{t+1}$ if we have $\bar{f}_t = f_t$ for all t. We trivially have $\bar{f}_0 = f_0$, so we will make the inductive assumption that $\bar{f}_{t-1} = f_{t-1}$ and show that this implies $\bar{f}_t = f_t$.

At each iteration $t \geq 1$, the direction $\mathbf{e}_k$ selected by coordinate descent is the one minimizing the directional derivative, which is defined as:

$$F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) = \lim_{\eta \to 0} \frac{F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) - F(\bar{\boldsymbol{\alpha}}_{t-1})}{\eta}\,.$$

Since $F(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k) = \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{t-1} \bar{\alpha}_{t-1,j}(h_j(x_i') - h_j(x_i)) - \eta y_i (h_k(x_i') - h_k(x_i))}$, the directional derivative along $\mathbf{e}_k$ can be expressed as follows:

$$\begin{aligned}
&F'(\bar{\boldsymbol{\alpha}}_{t-1}, \mathbf{e}_k) \\
&= -\sum_{i=1}^{m} y_i (h_k(x_i') - h_k(x_i)) \exp\Big[- y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j} (h_j(x_i') - h_j(x_i))\Big] \\
&= -\sum_{i=1}^{m} y_i (h_k(x_i') - h_k(x_i)) \bar{\mathcal{D}}_t(i) \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big] \\
&= -\Big[\sum_{i=1}^{m} \bar{\mathcal{D}}_t(i) 1_{y_i (h_t(x_i') - h_t(x_i)) = +1} - \sum_{i=1}^{m} \bar{\mathcal{D}}_t(i) 1_{y_i (h_t(x_i') - h_t(x_i)) = -1} \Big] \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big] \\
&= -[\bar{\epsilon}_t^{+} - \bar{\epsilon}_t^{-}] \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big].
\end{aligned}$$

The first equality holds by differentiation and evaluation at $\eta = 0$ and the second one follows from (10.12). In view of the final equality, since $m \prod_{s=1}^{t-1} \bar{Z}_s$ is fixed, the direction $\mathbf{e}_k$ selected by coordinate descent is the one minimizing $\bar{\epsilon}_t$. By the inductive hypothesis $\bar{\mathcal{D}}_t = \mathcal{D}_t$ and $\bar{\epsilon}_t = \epsilon_t$, thus, the chosen base ranker corresponds exactly to the base ranker h_t selected by RankBoost.

The step size η is identified by setting the derivative to zero in order to minimize the function in the chosen direction $\mathbf{e}_k$. Thus, using identity 10.12 and the definition

of $\bar{\epsilon}_t$, we can write:

$$\begin{aligned}
&\frac{dF(\bar{\boldsymbol{\alpha}}_{t-1} + \eta \mathbf{e}_k)}{d\eta} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i (h_t(x'_i) - h_t(x_i)) e^{-y_i \sum_{j=1}^{N} \bar{\alpha}_{t-1,j} (h_j(x'_i) - h_j(x_i))} e^{-\eta y_i (h_k(x'_i) - h_k(x_i))} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i (h_t(x'_i) - h_t(x_i)) \bar{\mathcal{D}}_t(i) \Big[m \prod_{s=1}^{t-1} \bar{Z}_s \Big] e^{-\eta y_i (h_k(x'_i) - h_k(x_i))} = 0 \\
&\Leftrightarrow -\sum_{i=1}^{m} y_i (h_t(x'_i) - h_t(x_i)) \bar{\mathcal{D}}_t(i)\, e^{-\eta y_i (h_t(x'_i) - h_t(x_i))} = 0 \\
&\Leftrightarrow -[\bar{\epsilon}_t^{+} e^{-\eta} - \bar{\epsilon}_t^{-} e^{\eta}] = 0 \\
&\Leftrightarrow \eta = \frac{1}{2} \log \frac{\bar{\epsilon}_t^{+}}{\bar{\epsilon}_t^{-}}.
\end{aligned}$$

By the inductive hypothesis, we have $\bar{\epsilon}_t^{+} = \epsilon_t^{+}$ and $\bar{\epsilon}_t^{-} = \epsilon_t^{-}$ and this proves that the step size chosen by coordinate descent matches the base ranker weight α_t of RankBoost. Thus, by combining the previous results we have $\bar{f}_t = f_t$ and the proof by induction is complete. This shows that coordinate descent applied to F precisely coincides with the RankBoost algorithm.

As in the classification case, other convex loss functions upper bounding the zero-one pairwise misranking loss can be used. In particular, the following objective function based on the logistic loss can be used: $\bar{\boldsymbol{\alpha}} \mapsto \sum_{i=1}^{m} \log(1 + e^{-y_i [f_N(x'_i) - f_N(x_i)]})$ to derive an alternative boosting-type algorithm.

10.4.3 Margin bound for ensemble methods in ranking

To simplify the presentation, we will assume for the results of this section, as in section 10.2, that the pairwise labels are in $\{-1, +1\}$. By lemma 7.4, the empirical Rademacher complexity of the convex hull $\operatorname{conv}(\mathcal{H})$ equals that of $\mathcal{H}$. Thus, theorem 10.1 immediately implies the following guarantee for ensembles of hypotheses in ranking.

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 $h \in \operatorname{conv}(\mathcal{H})$:*

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

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

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

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

Remarkably, the number of rounds of boosting T does not appear in this bound. The bound depends only on the margin ρ, the sample size m, and the Rademacher complexity of the family of base classifiers $\mathcal{H}$. Thus, the bound guarantees an effective generalization if the pairwise margin loss $\widehat{R}_{S,\rho}(f/\|\boldsymbol{\alpha}\|_1)$ is small for a relatively large ρ. A bound similar to that of theorem 7.7 for AdaBoost can be derived for the empirical pairwise ranking margin loss of RankBoost (see exercise 10.3) and similar comments on that result apply here.

These results provide a margin-based analysis in support of ensemble methods in ranking and RankBoost in particular. As in the case of AdaBoost, however, RankBoost in general does not achieve a maximum margin. But, in practice, it has been observed to obtain excellent pairwise ranking performances.

10.5 Bipartite ranking

This section examines an important ranking scenario within the score-based setting, the *bipartite ranking problem*. In this scenario, the set of points $\mathcal{X}$ is partitioned into two classes: $\mathcal{X}_+$ the class of positive points, and $\mathcal{X}_-$ that of negative ones. The problem consists of ranking positive points higher than negative ones. For example, for a fixed search engine query, the task consists of ranking relevant (positive) documents higher than irrelevant (negative) ones.

The bipartite problem could be treated in the way already discussed in the previous sections with exactly the same theory and algorithms. However, the setup typically adopted for this problem is different: instead of assuming that the learner receives a sample of random pairs, here pairs of positive and negative elements, it is assumed that it receives a sample of positive points from some distribution and a sample of negative points from another. This leads to the set of all pairs made of a positive point of the first sample and a negative point of the second.

More formally, the learner receives a sample $S_+ = (x'_1, \ldots, x'_m)$ drawn i.i.d. according to some distribution $\mathcal{D}_+$ over $\mathcal{X}_+$, and a sample $S_- = (x_1, \ldots, x_n)$ drawn i.i.d. according to some distribution $\mathcal{D}_-$ over $\mathcal{X}_-$.[17] Given a hypothesis set $\mathcal{H}$ of

[17] This two-distribution formulation also avoids a potential dependency issue that can arise for some modeling of the problem: if pairs are drawn according to some distribution $\mathcal{D}$ over $\mathcal{X}_- \times \mathcal{X}_+$

functions mapping $\mathcal{X}$ to $\mathbb{R}$, the learning problem consists of selecting a hypothesis $h \in \mathcal{H}$ with small expected bipartite misranking or generalization error $R(h)$:

$$R(h) = \mathop{\mathbb{P}}_{\substack{x \sim \mathcal{D}_- \\ x' \sim \mathcal{D}_+}} [h(x') < h(x)] \,. \tag{10.20}$$

The empirical pairwise misranking or empirical error of h is denoted by $\widehat{R}_{S_+,S_-}(h)$ and defined by

$$\widehat{R}_{S_+,S_-}(h) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} 1_{h(x'_i) < h(x_j)} \,. \tag{10.21}$$

Note that while the bipartite ranking problem bears some similarity with binary classification, in particular, the presence of two classes, they are distinct problems, since their objectives and measures of success clearly differ.

By the definition of the formulation of the bipartite ranking just presented, the learning algorithm must typically deal with mn pairs. For example, the application of SVMs to ranking in this scenario leads to an optimization with mn slack variables or constraints. With just a thousand positive and a thousand negative points, one million pairs would need to be considered. This can lead to a prohibitive computational cost for some learning algorithms. The next section shows that RankBoost admits an efficient implementation in the bipartite scenario.

10.5.1 Boosting in bipartite ranking

This section shows the efficiency of RankBoost in the bipartite scenario and discusses the connection between AdaBoost and RankBoost in this context.

The key property of RankBoost leading to an efficient algorithm in the bipartite setting is the fact that its objective function is based on the exponential function. As a result, it can be decomposed into the product of two functions, one depending on only the positive and the other on only the negative points. Similarly, the distribution $\mathcal{D}_t$ maintained by the algorithm can be factored as the product of two distributions $\mathcal{D}_t^+$ and $\mathcal{D}_t^-$. This is clear for the uniform distribution $\mathcal{D}_1$ at the first round as for any $i \in [m]$ and $j \in [n]$, $\mathcal{D}_1(i,j) = 1/(mn) = \mathcal{D}_1^+(i)\mathcal{D}_1^-(j)$ with $\mathcal{D}_1^+(i) = 1/m$ and $\mathcal{D}_1^-(j) = 1/n$. This property is recursively preserved since, in view of the following, the decomposition of $\mathcal{D}_t$ implies that of $\mathcal{D}_{t+1}$ for any $t \in [T]$.

---

and the learner makes use of this information to augment its training sample, then the resulting sample is in general not i.i.d. This is because if (x_1, x'_1) and (x_2, x'_2) are in the sample, then so are the pairs (x_1, x'_2) and (x_2, x'_1) and thus the pairs are not independent. However, without sample augmentation, the points are i.i.d., and this issue does not arise.

For any $i \in [m]$ and $j \in [n]$, by definition of the update, we can write:

$$\mathcal{D}_{t+1}(i,j) = \frac{\mathcal{D}_t(i,j)e^{-\alpha_t[h_t(x'_i)-h_t(x_j)]}}{Z_t} = \frac{\mathcal{D}_t^+(i)e^{-\alpha_t h_t(x'_i)}}{Z_{t,+}} \frac{\mathcal{D}_t^-(j)e^{\alpha_t h_t(x_j)}}{Z_{t,-}},$$

since the normalization factor Z_t can also be decomposed as $Z_t = Z_t^- Z_t^+$, with $Z_t^+ = \sum_{i=1}^m \mathcal{D}_t^+(i)e^{-\alpha_t h_t(x'_i)}$ and $Z_t^- = \sum_{j=1}^n \mathcal{D}_t^-(j)e^{\alpha_t h_t(x_j)}$. Furthermore, the pairwise misranking of a hypothesis $h \in \mathcal{H}$ based on the distribution $\mathcal{D}_t$ used to determine h_t can also be computed as the difference of two quantities, one depending only on positive points, the other only on negative ones:

$$\mathop{\mathbb{E}}_{(i,j)\sim\mathcal{D}_t}[h(x'_i) - h(x_j)] = \mathop{\mathbb{E}}_{i\sim\mathcal{D}_t^+}[\mathop{\mathbb{E}}_{j\sim\mathcal{D}_t^-}[h(x'_i) - h(x_j)]] = \mathop{\mathbb{E}}_{i\sim\mathcal{D}_t^+}[h(x'_i)] - \mathop{\mathbb{E}}_{j\sim\mathcal{D}_t^-}[h(x_j)].$$

Thus, the time and space complexity of RankBoost depends only on the total number of points $m+n$ and not the number of pairs mn. More specifically, ignoring the call to the weak ranker or the cost of determining h_t, the time and space complexity of each round is linear, that is, in $O(m+n)$. Furthermore, the cost of determining h_t is a function of $O(m+n)$ and not $O(mn)$. Figure 10.2 gives the pseudocode of the algorithm adapted to the bipartite scenario.

In the bipartite scenario, a connection can be made between the classification algorithm AdaBoost and the ranking algorithm RankBoost. In particular, the objective function of RankBoost can be expressed as follows for any $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T) \in \mathbb{R}^T$, $T \geq 1$:

$$\begin{aligned} F_{\text{RankBoost}}(\boldsymbol{\alpha}) &= \sum_{j=1}^m \sum_{i=1}^n \exp(-[f(x'_i) - f(x_j)]) \\ &= \Big(\sum_{i=1}^m e^{-\sum_{t=1}^T \alpha_t h_t(x'_i)}\Big)\Big(\sum_{j=1}^n e^{+\sum_{t=1}^T \alpha_t h_t(x_j)}\Big) \\ &= F_+(\boldsymbol{\alpha})F_-(\boldsymbol{\alpha}), \end{aligned}$$

where F_+ denotes the function defined by the sum over the positive points and F_- the function defined over the negative points. The objective function of AdaBoost can be defined in terms of these same two functions as follows:

$$\begin{aligned} F_{\text{AdaBoost}}(\boldsymbol{\alpha}) &= \sum_{i=1}^m \exp(-y'_i f(x'_i)) + \sum_{j=1}^n \exp(-y_j f(x_j)) \\ &= \sum_{i=1}^m e^{-\sum_{t=1}^T \alpha_t h_t(x'_i)} + \sum_{j=1}^n e^{+\sum_{t=1}^T \alpha_t h_t(x_j)} \\ &= F_+(\boldsymbol{\alpha}) + F_-(\boldsymbol{\alpha}). \end{aligned}$$

BIPARTITERANKBOOST($S = (x'_1, \ldots, x'_m, x_1, \ldots, x_n)$)

for $j \leftarrow 1$ **to** m **do**
  $\mathcal{D}_1^+(j) \leftarrow \frac{1}{m}$
for $i \leftarrow 1$ **to** n **do**
  $\mathcal{D}_1^-(i) \leftarrow \frac{1}{n}$
for $t \leftarrow 1$ **to** T **do**
  $h_t \leftarrow$ base ranker in $\mathcal{H}$ with smallest $\epsilon_t^- - \epsilon_t^+ = \mathop{\mathbb{E}}_{j \sim \mathcal{D}_t^-}[h(x_j)] - \mathop{\mathbb{E}}_{i \sim \mathcal{D}_t^+}[h(x'_i)]$
  $\alpha_t \leftarrow \frac{1}{2} \log \frac{\epsilon_t^+}{\epsilon_t^-}$
  $Z_t^+ \leftarrow 1 - \epsilon_t^+ + \sqrt{\epsilon_t^+ \epsilon_t^-}$
  for $i \leftarrow 1$ **to** m **do**
    $\mathcal{D}_{t+1}^+(i) \leftarrow \frac{\mathcal{D}_t^+(i) \exp\left[-\alpha_t h_t(x'_i)\right]}{Z_t^+}$
  $Z_t^- \leftarrow 1 - \epsilon_t^- + \sqrt{\epsilon_t^+ \epsilon_t^-}$
  for $j \leftarrow 1$ **to** n **do**
    $\mathcal{D}_{t+1}^-(j) \leftarrow \frac{\mathcal{D}_t^-(j) \exp\left[+\alpha_t h_t(x_j)\right]}{Z_t}$
$f \leftarrow \sum_{t=1}^T \alpha_t h_t$
return f

Figure 10.2
Pseudocode of RankBoost in a bipartite setting, with $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, $\epsilon_t^+ = \mathbb{E}_{i \sim \mathcal{D}_t^+}[h(x'_i)]$ and $\epsilon_t^- = \mathbb{E}_{j \sim \mathcal{D}_t^-}[h(x_j)]$.

Note that the gradient of the objective function of RankBoost can be expressed in terms of AdaBoost as follows:

$$\begin{aligned}\nabla_{\boldsymbol{\alpha}} F_{\text{RankBoost}}(\boldsymbol{\alpha}) &= F_-(\boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} F_+(\boldsymbol{\alpha}) + F_+(\boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha}) \qquad (10.22)\\ &= F_-(\boldsymbol{\alpha})(\nabla_{\boldsymbol{\alpha}} F_+(\boldsymbol{\alpha}) + \nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha})) + (F_+(\boldsymbol{\alpha}) - F_-(\boldsymbol{\alpha})) \nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha})\\ &= F_-(\boldsymbol{\alpha}) \nabla_{\boldsymbol{\alpha}} F_{\text{AdaBoost}}(\boldsymbol{\alpha}) + (F_+(\boldsymbol{\alpha}) - F_-(\boldsymbol{\alpha})) \nabla_{\boldsymbol{\alpha}} F_-(\boldsymbol{\alpha}).\end{aligned}$$

If $\boldsymbol{\alpha}$ is a minimizer of F_{AdaBoost}, then $\nabla_{\boldsymbol{\alpha}} F_{\text{AdaBoost}}(\boldsymbol{\alpha}) = 0$ and it can be shown that the equality $F_+(\boldsymbol{\alpha}) - F_-(\boldsymbol{\alpha}) = 0$ also holds for $\boldsymbol{\alpha}$, provided that the family of base hypotheses $\mathcal{H}$ used for AdaBoost includes the constant hypothesis $h_0 \colon x \mapsto 1$, which often is the case in practice. Then, by (10.22), this implies that $\nabla_{\boldsymbol{\alpha}} F_{\text{RankBoost}}(\boldsymbol{\alpha}) = 0$ and therefore that $\boldsymbol{\alpha}$ is also a minimizer of the convex

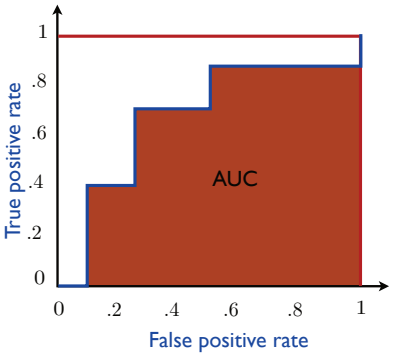


Figure 10.3
The AUC (area under the ROC curve) is a measure of the performance of a bipartite ranking.

function $F_{\text{RankBoost}}$. In general, F_{AdaBoost} does not admit a minimizer. Nevertheless, it can be shown that if $\lim_{k\to\infty} F_{\text{AdaBoost}}(\boldsymbol{\alpha}_k) = \inf_{\boldsymbol{\alpha}} F_{\text{AdaBoost}}(\boldsymbol{\alpha})$ for some sequence $(\boldsymbol{\alpha}_k)_{k\in\mathbb{N}}$, then, under the same assumption on the use of a constant base hypothesis and for a non-linearly separable dataset, the following holds: $\lim_{k\to\infty} F_{\text{RankBoost}}(\boldsymbol{\alpha}_k) = \inf_{\boldsymbol{\alpha}} F_{\text{RankBoost}}(\boldsymbol{\alpha})$.

The connections between AdaBoost and RankBoost just mentioned suggest that AdaBoost could achieve a good ranking performance as well. This is often observed empirically, a fact that brings strong support to the use of AdaBoost both as a classifier and a ranking algorithm. Nevertheless, RankBoost may converge faster and achieve a good ranking faster than AdaBoost.

10.5.2 Area under the ROC curve

The performance of a bipartite ranking algorithm is typically reported in terms of the *area under the receiver operating characteristic (ROC) curve*, or the *area under the curve* (*AUC*) for short.

Let U be a test sample used to evaluate the performance of h (or a training sample) with m positive points $z'_1, \ldots, z'_m$ and n negative points $z_1, \ldots, z_n$. For any $h \in \mathcal{H}$, let $\widehat{R}(h, U)$ denote the average pairwise misranking of h over U. Then, the AUC of h for the sample U is precisely $1 - \widehat{R}(h, U)$, that is, its average pairwise ranking accuracy on U:

$$\text{AUC}(h, U) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} 1_{h(z'_i) \geq h(z_j)} = \mathop{\mathbb{P}}_{\substack{z \sim \widehat{\mathcal{D}}_U^- \\ z' \sim \widehat{\mathcal{D}}_U^+}} [h(z') \geq h(z)].$$

Here, $\widehat{\mathcal{D}}_U^+$ denotes the empirical distribution corresponding to the positive points in U and $\widehat{\mathcal{D}}_U^+$ the empirical distribution corresponding to the negative ones. $\text{AUC}(h, U)$

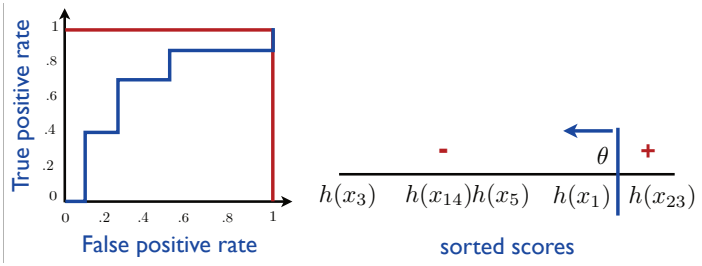


Figure 10.4
An example ROC curve and illustrated threshold. Varying the value of θ from one extreme to the other generates points on the curve.

is thus an empirical estimate of the pairwise ranking accuracy based on the sample U, and by definition it is in $[0, 1]$. Higher AUC values correspond to a better ranking performance. In particular, an AUC of one indicates that the points of U are ranked perfectly using h. $\mathrm{AUC}(h, U)$ can be computed in linear time from a sorted array containing the $m+n$ elements $h(z'_i)$ and $h(z_j)$, for $i \in [m]$ and $j \in [n]$. Assuming that the array is sorted in increasing order (with a positive point placed higher than a negative one if they both have the same scores) the total number of correctly ranked pairs r can be computed as follows. Starting with $r = 0$, the array is inspected in increasing order of the indices while maintaining at any time the number of negative points seen n and incrementing the current value of r with n whenever a positive point is found. After full inspection of the array, the AUC is given by $r/(mn)$. Thus, assuming that a comparison-based sorting algorithm is used, the complexity of the computation of the AUC is in $O((m+n)\log(m+n))$.

As indicated by its name, the AUC coincides with the area under the ROC curve (figure 10.3). An ROC curve plots the *true positive rate*, that is, the percentage of positive points correctly predicted as positive as a function of the *false positive rate*, that is, the percentage of negative points incorrectly predicted as positive. Figure 10.4 illustrates the definition and construction of an ROC curve.

Points are generated along the curve by varying a threshold value θ as in the right panel of figure 10.4, from higher values to lower ones. The threshold is used to determine the label of any point x (positive or negative) based on $\mathrm{sgn}(h(x) - \theta)$. At one extreme, all points are predicted as negative; thus, the false positive rate is zero, but the true positive rate is zero as well. This gives the first point $(0, 0)$ of the plot. At the other extreme, all points are predicted as positive; thus, both the true and the false positive rates are equal to one, which gives the point $(1, 1)$. In the ideal case, as already discussed, the AUC value is one, and, with the exception of $(0, 0)$, the curve coincides with a horizontal line reaching $(1, 1)$.

10.6 Preference-based setting

This section examines a different setting for the problem of learning to rank: the *preference-based setting*. In this setting, the objective is to rank as accurately as possible any test subset $X \subseteq \mathcal{X}$, typically a finite set that we refer to as a *finite query subset*. This is close to the query-based scenario of search engines or information extraction systems and the terminology stems from the fact that X could be a set of items needed to rank in response to a particular query. The advantage of this setting over the score-based setting is that here the learning algorithm is not required to return a linear ordering of all points of $\mathcal{X}$, which may be impossible to achieve faultlessly in accordance with a general possibly non-transitive pairwise preference labeling. Supplying a correct linear ordering for a query subset is more likely to be achievable exactly or at least with a better approximation.

The preference-based setting consists of two stages. In the first stage, a sample of labeled pairs S, exactly as in the score-based setting, is used to learn a *preference function* $h : \mathcal{X} \times \mathcal{X} \mapsto [0, 1]$, that is, a function that assigns a higher value to a pair (u, v) when u is preferred to v or is to be ranked higher than v, and smaller values in the opposite case. This preference function can be obtained as the output of a standard classification algorithm trained on S. A crucial difference with the score-based setting is that, in general, the preference function h is not required to induce a linear ordering. The relation it induces may be non-transitive; thus, we may have, for example, $h(u, v) = h(v, w) = h(w, u) = 1$ for three distinct points u, v, and w.

In the second stage, given a query subset $X \subseteq \mathcal{X}$, the preference function h is used to determine a ranking of X. How can h be used to generate an accurate ranking? This will be the main focus of this section. The computational complexity of the algorithm determining the ranking is also crucial. Here, we will measure its running time complexity in terms of the number of calls to h.

When the preference function is obtained as the output of a binary classification algorithm, the preference-based setting can be viewed as a reduction of ranking to classification: the second stage specifies how a ranking is obtained from a classifier's output.

10.6.1 Second-stage ranking problem

The ranking problem of the second stage is modeled as follows. We assume that a preference function h is given. From the point of view of this stage, the way the function h has been determined is immaterial, it can be viewed as a black box. As already discussed, h is not assumed to be transitive. But, we will assume that it is *pairwise consistent*, that is $h(u, v) + h(v, u) = 1$, for all $u, v \in \mathcal{X}$.

Let $\mathcal{D}$ be an unknown distribution according to which pairs (X, σ^*) are drawn where $X \subseteq \mathcal{X}$ is a query subset and σ^* a target ranking or permutation of X, that is, a bijective function from X to $\{1, \ldots, |X|\}$. Thus, we consider a stochastic scenario, and σ^* is a random variable. The objective of a second-stage algorithm $\mathcal{A}$ consists of using the preference function h to return an accurate ranking $\mathcal{A}(X)$ for any query subset X. The algorithm may be deterministic, in which case $\mathcal{A}(X)$ is uniquely determined from X or it may be randomized, in which case we denote by s the randomization seed it may depend on.

The following loss function L can be used to measure the disagreement between a ranking σ and a desired one σ^* over a set X of $n \geq 1$ elements:

$$L(\sigma, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} 1_{\sigma(u) < \sigma(v)} 1_{\sigma^*(v) < \sigma^*(u)}, \tag{10.23}$$

where the sum runs over all pairs (u, v) with u and v distinct elements of X. All the results presented in the following hold for a broader set of loss functions described later. Abusing the notation, we also define the loss of the preference function h with respect to a ranking σ^* of a set X of $n \geq 1$ elements by

$$L(h, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} h(u, v) 1_{\sigma^*(v) < \sigma^*(u)}. \tag{10.24}$$

The expected loss for a deterministic algorithm $\mathcal{A}$ is thus $\mathbb{E}_{(X,\sigma^*)\sim\mathcal{D}}[L(\mathcal{A}(X), \sigma^*)]$. The *regret* of algorithm $\mathcal{A}$ is then defined as the difference between its loss and that of the best fixed global ranking. This can be written as follows:

$$\mathrm{Reg}(\mathcal{A}) = \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(\mathcal{A}(X), \sigma^*)] - \min_{\sigma'} \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(\sigma'_{|X}, \sigma^*)]\,, \tag{10.25}$$

where $\sigma'_{|X}$ denotes the ranking induced on X by a global ranking σ' of $\mathcal{X}$. Similarly, we define the regret of the preference function as follows

$$\mathrm{Reg}(h) = \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(h_{|X}, \sigma^*)] - \min_{h'} \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(h'_{|X}, \sigma^*)]\,, \tag{10.26}$$

where $h_{|X}$ denotes the restriction of h to $X \times X$, and similarly with h'. The regret results presented in this section hold assuming the following *pairwise independence on irrelevant alternatives* property:

$$\mathop{\mathbb{E}}_{\sigma^*|X_1}[1_{\sigma^*(v) < \sigma^*(u)}] = \mathop{\mathbb{E}}_{\sigma^*|X_2}[1_{\sigma^*(v) < \sigma^*(u)}], \tag{10.27}$$

for any $u, v \in \mathcal{X}$ and any two sets X_1 and X_2 containing u and v, and where $\sigma^*|X$ denotes the random variable σ^* conditioned on X.[18] Similar regret definitions can be given for a randomized algorithm additionally taking the expectation over s.

Clearly, the quality of the ranking output by the second-stage algorithm intimately depends on that of the preference function h. In the next sections, we discuss both a deterministic and a randomized second-stage algorithm for which the regret can be upper bounded in terms of the regret of the preference function.

10.6.2 Deterministic algorithm

A natural deterministic algorithm for the second-stage is based on the *sort-by-degree algorithm*. This consists of ranking each element of X based on the number of other elements it is preferred to according to the preference function h. Let $\mathcal{A}_{\text{sort-by-degree}}$ denote this algorithm. In the bipartite setting, the following bounds can be proven for the expected loss of this algorithm and its regret:

$$\mathop{\mathbb{E}}_{X,\sigma^*}[L(\mathcal{A}_{\text{sort-by-degree}}(X), \sigma^*)] \leq 2 \mathop{\mathbb{E}}_{X,\sigma^*}[L(h, \sigma^*)] \tag{10.30}$$

$$\text{Reg}(\mathcal{A}_{\text{sort-by-degree}}(X)) \leq 2\,\text{Reg}(h)\,. \tag{10.31}$$

These results show that the sort-by-degree algorithm can achieve an accurate ranking when the loss or the regret of the preference function h is small. They also bound the *ranking* loss or regret of the algorithm in terms of the *classification* loss or regret of h, which can be viewed as a guarantee for the reduction of ranking to classification using the sort-by-degree algorithm.

Nevertheless, in some cases, the guarantee given by these results is weak or uninformative owing to the presence of the factor of two. Consider the case of a binary classifier h with an error rate of just 25 percent, which is quite reasonable in many applications. Assume that the Bayes error is close to zero for the classification problem and, similarly, that for the ranking problem the regret and loss approximately coincide. Then, using the bound in (10.30) guarantees a worst-case pairwise misranking error of at most 50 percent for the ranking algorithm, which is the pairwise misranking error of random ranking.

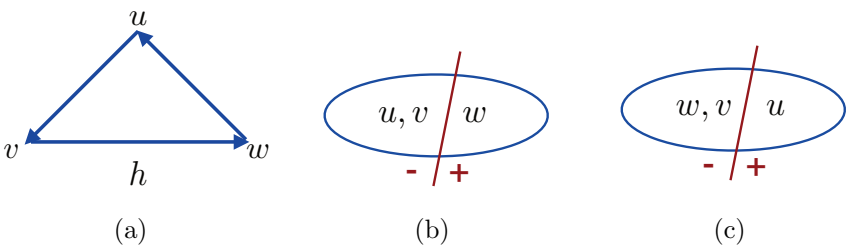
---

[18] More generally, they hold without that assumption using the following weaker notions of regret:

$$\text{Reg}'(\mathcal{A}) = \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(\mathcal{A}(X), \sigma^*)] - \mathop{\mathbb{E}}_{X}\left[\min_{\sigma'} \mathop{\mathbb{E}}_{\sigma^*|X}[L(\sigma', \sigma^*)]\right] \tag{10.28}$$

$$\text{Reg}'(h) = \mathop{\mathbb{E}}_{(X,\sigma^*)\sim\mathcal{D}}[L(h_{|X}, \sigma^*)] - \mathop{\mathbb{E}}_{X}\left[\min_{h'} \mathop{\mathbb{E}}_{\sigma^*|X}[L(h', \sigma^*)]\right], \tag{10.29}$$

where σ' denotes a ranking of X and h' a preference function defined over $X \times X$.



(a) (b) (c)

Figure 10.5
Illustration of the proof of theorem 10.5.

Furthermore, the running time complexity of the algorithm quadratic, that is in $\Omega(|X|^2)$ of a query set X, since it requires calling the preference function for every pair (u, v) with u and v in X.

As shown by the following theorem, no deterministic algorithm can improve upon the factor of two appearing in the regret guarantee of the sort-by-degree algorithm.

Theorem 10.5 (Lower bound for deterministic algorithms) *For any deterministic algorithm $\mathcal{A}$, there is a bipartite distribution for which*

$$\mathrm{Reg}(\mathcal{A}) \geq 2\,\mathrm{Reg}(h). \tag{10.32}$$

Proof: Consider the simple case where $\mathcal{X} = X = \{u, v, w\}$ and where the preference function induces a cycle as illustrated by figure 10.5a. An arrow from u to v indicates that v is preferred to u according to h. The proof is based on an adversarial choice of the target σ^*.

Without loss of generality, either $\mathcal{A}$ returns the ranking u, v, w (figure 10.5b) or w, v, u (figure 10.5c). In the first case, let σ^* be defined by the labeling indicated in the figure. In that case, we have $L(h, \sigma^*) = 1/3$, since u is preferred to w according to h while w is labeled positively and u negatively. The loss of the algorithm is $L(\mathcal{A}, \sigma^*) = 2/3$, since both u and v are ranked higher than the positively labeled w by the algorithm. Similarly, σ^* can be defined as in figure 10.5c in the second case, and we find again that $L(h, \sigma^*) = 1/3$ and $L(\mathcal{A}, \sigma^*) = 2/3$. This concludes the proof. □

The theorem suggests that randomization is necessary in order to achieve a better guarantee. In the next section, we present a randomized algorithm that benefits both from better guarantees and a better time complexity.

10.6.3 Randomized algorithm

The general idea of the algorithm described in this section is to use a straightforward extension of the randomized QuickSort algorithm in the second stage. Unlike in

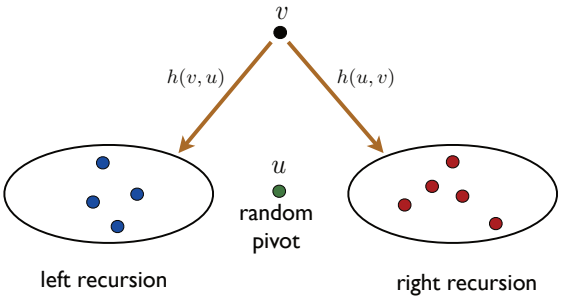


Figure 10.6
Illustration of randomized QuickSort based on a preference function h (not necessarily transitive).

the standard version of QuickSort, here the comparison function is based on the preference function, which in general is not transitive. Nevertheless, it can be shown here, too, that the expected time complexity of the algorithm is in $O(n \log n)$ when applied to an array of size n.

The algorithm works as follows, as illustrated by figure 10.6. At each recursive step, a *pivot* element u is selected uniformly at random from X. For each $v \neq u$, v is placed on the left of u with probability $h(v, u)$ and to its right with the remaining probability $h(u, v)$. The algorithm proceeds recursively with the array to the left of u and the one to its right and returns the concatenation of the permutation returned by the left recursion, u, and the permutation returned by the right recursion.

Let $\mathcal{A}_{\text{QuickSort}}$ denote this algorithm. In the bipartite setting, the following guarantees can be proven:

$$\mathop{\mathbb{E}}_{X,\sigma^*,s}[L(\mathcal{A}_{\text{QuickSort}}(X, s), \sigma^*)] = \mathop{\mathbb{E}}_{X,\sigma^*}[L(h, \sigma^*)] \tag{10.33}$$

$$\text{Reg}(\mathcal{A}_{\text{QuickSort}}) \leq \text{Reg}(h)\,. \tag{10.34}$$

Thus, here, the factor of two of the bounds in the deterministic case has vanished, which is substantially more favorable. Furthermore, the guarantee for the loss is an equality. Moreover, the expected time complexity of the algorithm is only in $O(n \log n)$, and, if only the top k items are needed to be ranked, as in many applications, the time complexity is reduced to $O(n + k \log k)$.

For the QuickSort algorithm, the following guarantee can also be proven in the case of general ranking setting (not necessarily bipartite setting):

$$\mathop{\mathbb{E}}_{X,\sigma^*,s}[L(\mathcal{A}_{\text{QuickSort}}(X, s), \sigma^*)] \leq 2 \mathop{\mathbb{E}}_{X,\sigma^*}[L(h, \sigma^*)]. \tag{10.35}$$

10.6.4 Extension to other loss functions

All of the results just presented hold for a broader class of loss functions L_ω defined in terms of a *weight function* or *emphasis function* ω. L_ω is similar to (10.23), but measures the weighted disagreement between a ranking σ and a desired one σ^* over a set X of $n \geq 1$ elements as follows:

$$L_\omega(\sigma, \sigma^*) = \frac{2}{n(n-1)} \sum_{u \neq v} \omega(\sigma^*(v), \sigma^*(u))\, 1_{\sigma(u)<\sigma(v)}\, 1_{\sigma^*(v)<\sigma^*(u)}, \tag{10.36}$$

where the sum runs over all pairs (u, v) with u and v distinct elements of X, and where ω is a symmetric function whose properties are described below. Thus, the loss counts the number of pairwise misrankings of σ with respect to σ^*, each weighted by ω. The function ω is assumed to satisfy the following three natural axioms:

- 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: $\omega(i, j) \leq \omega(i, k) + \omega(k, j)$.

The motivation for this last property stems from the following: if correctly ordering items in positions (i, k) and (k, j) is not of great importance, then the same should hold for items in positions (i, j).

Using different functions ω, the family of functions L_ω can cover several familiar and important losses. Here are some examples. Setting $\omega(i, j) = 1$ for all $i \neq j$ yields the unweighted pairwise misranking measure. For a fixed integer $k \geq 1$, the function ω defined by $\omega(i, j) = 1_{((i \leq k) \vee (j \leq k)) \wedge (i \neq j)}$ for all (i, j) can be used to emphasize ranking at the top k elements. Misranking of pairs with at least one element ranked among the top k is penalized by this function. This can be of interest in applications such as information extraction or search engines where the ranking of the top documents matters more. For this emphasis function, all elements ranked below k are in a tie. Any tie relation can be encoded using ω. Finally, in a bipartite ranking scenario with m^+ positive and m^- negative points and $m^+ + m^- = n$, choosing $\omega(i, j) = \frac{n(n-1)}{2m^- m^+}$ yields the standard loss function coinciding with $1 - \text{AUC}$.

10.7 Other ranking criteria

The objective function for the ranking problems discussed in this chapter were all based on pairwise misranking. Other ranking criteria have been introduced in information retrieval and used to derive alternative ranking algorithms. Here, we briefly present several of these criteria.

- Precision, precision@n, average precision, recall. All of these criteria assume that points are partitioned into two classes (positives and negatives), as in the bipartite ranking setting. *Precision* is the fraction of positively predicted points that are in fact positive. Whereas precision takes into account all positive predictions, *precision@n* only considers the top n predictions. For example, precision@5 considers only the top 5 positively predicted points. *Average precision* involves computing precision@n for each value of n, and averaging across these values. Each precision@n computation can be interpreted as computing precision for a fixed value of *recall*, or the fraction of positive points that are predicted to be positive (recall coincides with the notion of true positive rate).
- DCG, NDCG. These criteria assume the existence of relevance scores associated with the points to be ranked, e.g., given a web search query, each website returned by a search engine has an associated relevance score. Moreover, these criteria measure the extent to which points with large relevance scores appear at or near the beginning of a ranking. Define $(c_i)_{i \in \mathbb{N}}$ as a predefined sequence of non-increasing and non-negative discount factors, e.g., $c_i = \log(i)^{-1}$. Then, given a ranking of m points and defining r_i as the relevance score of the ith point in this ranking, the *discounted cumulative gain (DCG)* is defined as $DCG = \sum_{i=1}^{m} c_i r_i$. Note that DCG is an increasing function of m. In contrast, the *normalized discounted cumulative gain (NDCG)* normalizes the DCG across values of m by dividing the DCG by the IDCG, or the ideal DCG that would result from an optimal ordering of the points.

10.8 Chapter notes

The problem of learning to rank is distinct from the purely algorithmic one of rank aggregation, which, as shown by Dwork, Kumar, Naor, and Sivakumar [2001], is NP-hard even for $k = 4$ rankings. The Rademacher complexity and margin-based generalization bounds for pairwise ranking given in theorem 10.1 and corollary 6.13 are novel. Margin bounds based on covering numbers were also given by Rudin, Cortes, Mohri, and Schapire [2005]. Other learning bounds in the score-based setting of ranking, including VC-dimension and stability-based learning bounds, have been given by Agarwal and Niyogi [2005], Agarwal et al. [2005] and Cortes et al. [2007b].

The ranking algorithm based on SVMs presented in section 10.3 has been used and discussed by several researchers. One early and specific discussion of its use can be found in Joachims [2002]. The fact that the algorithm is simply a special instance of SVMs seems not to be clearly stated in the literature. The theoretical justification presented here for its use in ranking is novel.

RankBoost was introduced by Freund et al. [2003]. The version of the algorithm presented here is the coordinate descent RankBoost from Rudin et al. [2005]. RankBoost in general does not achieve a maximum margin and may not increase the margin at each iteration. A Smooth Margin ranking algorithm [Rudin et al., 2005] based on a modified version of the objective function of RankBoost can be shown to increase the smooth margin at every iteration, but the comparison of its empirical performance with that of RankBoost has not been reported. For the empirical ranking quality of AdaBoost and the connections between AdaBoost and RankBoost in the bipartite setting, see Cortes and Mohri [2003] and Rudin et al. [2005].

The Receiver Operating Characteristics (ROC) curves were originally developed in signal detection theory [Egan, 1975] in connection with radio signals during World War II. They also had applications to psychophysics [Green and Swets, 1966] and have been used since then in a variety of other applications, in particular for medical decision making. The area under an ROC curve (AUC) is equivalent to the Wilcoxon-Mann-Whitney statistic [Hanley and McNeil, 1982] and is closely related to the Gini index [Breiman et al., 1984] (see also chapter 9). For a statistical analysis of the AUC and confidence intervals depending on the error rate, see Cortes and Mohri [2003, 2005]. The deterministic algorithm in the preference-based setting discussed in this chapter was presented and analyzed by Balcan et al. [2008]. The randomized algorithm as well as much of the results presented in section 10.6 are due to Ailon and Mohri [2008].

A somewhat related problem of *ordinal regression* has been studied by some authors [McCullagh, 1980, McCullagh and Nelder, 1983, Herbrich et al., 2000] which consists of predicting the correct label of each item out of a finite set, as in multi-class classification, with the additional assumption of an ordering among the labels. This problem is distinct, however, from the pairwise ranking problem discussed in this chapter.

The DCG ranking criterion was introduced by Järvelin and Kekäläinen [2000], and has been used and discussed in a number of subsequent studies, in particular Cossock and Zhang [2008] who consider a subset ranking problem formulated in terms of DCG, for which they consider a regression-based solution.

10.9 Exercises

10.1 Uniform margin-bound for ranking. Use theorem 10.1 to derive a margin-based learning bound for ranking that holds uniformly for all $\rho > 0$ (see similar binary classification bounds of theorem 5.9 and exercise 5.2).

10.2 On-line ranking. Give an on-line version of the SVM-based ranking algorithm presented in section 10.3.

10.3 Empirical margin loss of RankBoost. Derive an upper bound on the empirical pairwise ranking margin loss of RankBoost similar to that of theorem 7.7 for AdaBoost.

10.4 Margin maximization and RankBoost. Give an example showing that RankBoost does not achieve the maximum margin, as in the case of AdaBoost.

10.5 RankPerceptron. Adapt the Perceptron algorithm to derive a pairwise ranking algorithm based on a linear scoring function. Assume that the training sample is linear separable for pairwise ranking. Give an upper bound on the number of updates made by the algorithm in terms of the ranking margin.

10.6 Margin-maximization ranking. Give a linear programming (LP) algorithm returning a linear hypothesis for pairwise ranking based on margin maximization.

10.7 Bipartite ranking. Suppose that we use a binary classifier for ranking in the bipartite setting. Prove that if the error of the binary classifier is ϵ, then that of the ranking it induces is also at most ϵ. Show that the converse does not hold.

10.8 Multipartite ranking. Consider the ranking scenario in a k-partite setting where $\mathcal{X}$ is partitioned into k subsets $\mathcal{X}_1, \ldots, \mathcal{X}_k$ with $k \geq 1$. The bipartite case ($k = 2$) is already specifically examined in the chapter. Give a precise formulation of the problem in terms of k distributions. Does RankBoost admit an efficient implementation in this case? Give the pseudocode of the algorithm.

10.9 Deviation bound for the AUC. Let h be a fixed scoring function used to rank the points of $\mathcal{X}$. Use Hoeffding's bound to show that with high probability the AUC of h for a finite sample is close to its average.

10.10 k-partite weight function. Show how the weight function ω can be defined so that L_ω encodes the natural loss function associated to a k-partite ranking scenario.