Efficient Learning of Mahalanobis Metrics for Ranking

Daryl K. H. Lim DKLIM@UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 USA

Gert Lanckriet GERT@ECE.UCSD.EDU
Department of Electrical and Computer Engineering, University of California, San Diego, CA 92093 USA

Abstract

We develop an efficient algorithm to learn a Mahalanobis distance metric by directly optimizing a ranking loss. Our approach focuses on optimizing the top of the induced rankings, which is desirable in tasks such as visualization and nearest-neighbor retrieval. We further develop and justify a simple technique to reduce training time significantly with minimal impact on performance. Our proposed method significantly outperforms alternative methods on several real-world tasks, and can scale to large and high-dimensional data.

1. Introduction

Distance metric learning algorithms learn a (linear) transformation optimized to yield small distances between *similar* pairs of points, and large distances between *dissimilar* pairs of points (Xing et al., 2003; Davis et al., 2007; Weinberger & Saul, 2009). Given a linear transformation L, the squared Euclidean distance between two points under the transformation can be written as $d(x, y) = \|Lx - Ly\|_2^2$, or

$$d(x, y) = \|x - y\|_W^2 = (x - y)^\mathsf{T} W (x - y)$$

where $W = L^\mathsf{T} L \succeq 0$ is a Mahalanobis metric. Our goal then is to optimize W such that the distances induced by W between points in the training set satisfy some constraints derived from the labels or available side information.

In many applications of metric learning, e.g. visualization or k-nearest-neighbor classification, we are usually interested in *local similarity*: given a test point, we want to be able to retrieve similar points by searching in a small neighborhood of the query. At a high level, a metric W is considered to be good if, given a test point, sorting the training set in increasing order of distance under W results in similar points appearing at the top of the ranking. A good measure of the quality of rankings induced by W is the Mean Average Precision (MAP). It is a popular choice in retrieval applications as it provides a stable measure of retrieval quality across multiple recall levels and is *top-heavy*, i.e., it rewards good performance at the top of the ranking, which is well-suited to our notion of local similarity.

To optimize for ranking performance directly, (McFee & Lanckriet, 2010) proposed the Metric Learning to Rank (MLR) algorithm, which is based on the structural SVM and optimized via the 1-Slack cutting plane algorithm. MLR has shown good performance in various ranking and classification tasks, and is able to optimize for a variety of ranking losses. However, despite the use of an efficient convex algorithm to solve the dual problem (Lim et al., 2013), the MLR algorithm has two drawbacks which limit its scalability to large, high-dimensional data sets: Firstly, at each iteration, a spectral decomposition of the metric must be performed to ensure that W is positive definite, which scales as $O(d^3)$, where d is the dimension of the data. Secondly, the constraint generation step can be expensive for listwise losses; the MAP separation oracle (Yue et al., 2007) can have complexity quadratic in the number of training examples.

Recently, a number of scalable methods (Chechik et al., 2009; Shalit et al., 2012) have been proposed to optimize a metric for a ranking loss. However, the ranking loss considered is these methods is usually the Area Under the ROC Curve (AUC) loss, which does not focus on the top of the list in general. Thus, there is a need for a scalable algorithm that optimizes a Mahalanobis metric with respect to a top-heavy ranking measure.

In this work, we propose an efficient distance metric learning algorithm for ranking which scales to high-dimensional and large datasets. Our approach combines recent Riemannian manifold optimization techniques with the recently proposed WARP loss to optimize the top of the ranking, and can be optimized via stochastic gradient descent (SGD). We also propose an extension to the sampling scheme for the WARP loss, and show that it can lead to

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

large speedups with minimal impact on performance. Our proposed method outperforms existing algorithms in terms of ranking performance on a number of real-world retrieval tasks, with significantly reduced computation time.

1.1. Preliminaries

Let $S^+_{d,m}$ denote the set of $d \times d$ symmetric positive semidefinite (PSD) matrices of rank m. Let $\langle A, B\rangle$ be the Frobenius inner product $:= \mathrm{tr}\,(A^\mathsf{T}B)$, and $\|A\|_\mathsf{F}$ be the Frobenius norm $:= \sqrt{\langle A, A\rangle}$. Given a PSD matrix W, let $\|x\|_W = \sqrt{x^\mathsf{T}Wx}$ be the weighted norm of x under W. Let $|\mathcal{X}|$ denote the cardinality of the set $\mathcal{X}$. Let $[\cdot]_+ := \max(0, \cdot)$. Let $\mathbf{I}[x] = 1$ if $x > 0$, and 0 otherwise.

2. Ranking via Metric Learning

In this paper, we propose an efficient distance metric learning algorithm that optimizes a Mahalanobis metric with respect to a top-heavy ranking loss. Before describing our formulation, we first describe two design choices that we made. To ensure scalability to high-dimensional data, we restrict W to $S^+_{d,m}$, where d is the dimension of the data, instead of optimizing W over the positive semidefinite cone. This results in large computational savings: in Section 3, we derive an algorithm to learn W which scales as $O(dm^2)$, which for small m is much more scalable than many current metric learning approaches which usually scale as $O(d^2)$ or $O(d^3)$. This design choice comes with the additional benefits of lower memory consumption during optimization, and the ability to perform dimensionality reduction of the training set with the learned metric.

To scale to large datasets, our algorithm is expressed as a sum of pairwise losses and can thus be optimized via stochastic gradient descent, which generally exhibits faster convergence than batch descent on large training sets.

Our formulation, Fast Ranking via Metric Learning (FRML) can be written as follows:

$$\min_{\substack{W\succeq 0\\ \mathrm{rank}(W)=m}} \sum_{q\in\mathcal{X}} \sum_{x^+\in\mathcal{X}_q^+} \mathcal{L}(r_q(x^+)) + \lambda\Omega(W) \qquad (1)$$

where

$$f_q(x) = -\|q - x\|^2_W$$
$$r_q(x^+) = \sum_{x^-\in\mathcal{X}_q^-} \mathbf{I}[f_q(x^-) - f_q(x^+)]$$

Here, $W \in \mathbb{R}^{d\times d}$ is the metric we wish to learn; $\mathcal{X} \subset \mathbb{R}^d$ is the training set of n points of dimension d; $q \in \mathcal{X}$ is a query with relevant set $\mathcal{X}_q^+ \subseteq \mathcal{X}$ and irrelevant set $\mathcal{X}_q^- \subseteq \mathcal{X}$; $r_q(x^+)$ is the *rank* of x^+, which we define as the number of points in $\mathcal{X}_q^-$ closer to q than x^+ is; and $\Omega(W)$ is a regularizer on W, e.g. $\Omega(W) = \|W\|^2_F$.

Inspired by the use of a similar term in (Weinberger & Saul, 2009), we set $\Omega(W)$ to be

$$\Omega_L(W) = \sum_{q\in\mathcal{X},\ x^+\in\mathcal{X}_q^+} \|q - x^+\|^2_W = -\sum_{q\in\mathcal{X},\ x^+\in\mathcal{X}_q^+} f_q(x^+) \qquad (2)$$

as it decomposes over (q, x^+) to give low-rank sample gradients, a benefit which will be made clear in Section 3.

$\mathcal{L} : \mathbb{Z}^+ \to \mathbb{R}^+$ is a mapping that transforms $r_q(x^+)$ into a loss. When $\mathcal{L}(\cdot)$ is set appropriately, FMRL optimizes a WARP loss (Weston et al., 2010), which we review below.

2.1. Weighted Approximate Pairwise Ranking (WARP)

The WARP loss is defined as follows:

$$l_{\text{WARP}}(q, f_q) = \frac{1}{|\mathcal{X}_q^+|} \sum_{x^+\in\mathcal{X}_q^+} \frac{1}{\mathcal{L}(|\mathcal{X}_q^-|)} \mathcal{L}(r_q(x^+))$$
$$\mathcal{L}(k) = \sum_{i=1}^{k} \alpha_i,\ \alpha_1 \geq \alpha_2 \cdots \alpha_{|\mathcal{X}_q^-|} \geq 0 \qquad (3)$$

where $\mathcal{L}(\cdot)$ is a mapping function which transforms the rank into a loss. Different choices of α for $\mathcal{L}(\cdot)$ lead to different minimizers: Setting all αs to be equal minimizes the mean rank, and larger values of α in the first few positions favor top-heavy rankings. In this paper we use $\alpha_i = \frac{1}{i}$, which has has shown good MAP and precision-at-k performance in (Weston et al., 2010).

$\mathcal{L}(|\mathcal{X}_q^-|)$ is a normalizer such that the worst ranking has a loss of 1. For simplicity, we assume for the rest of this paper that $|\mathcal{X}_q^-|$ is constant for all q, omitting the normalization term in further treatment.

As $\mathcal{L}(r_q(x^+))$ is discontinuous, we replace it with the continuous upper bound

$$\sum_{x^-\in\mathcal{V}_{q,x^+}} \mathcal{L}(r_q^m(x^+)) \frac{[1 - f_q(x^+) + f_q(x^-)]_+}{r_q^m(x^+)} \qquad (4)$$

where $\mathcal{V}_{q,x^+}$ is the set of *violators* for a given (q, x^+) pair:

$$\mathcal{V}_{q,x^+} = \{x^- \in \mathcal{X}_q^- : f_q(x^+) - f_q(x^-) < 1\}$$

and $r_q^m(x^+) = |\mathcal{V}_{q,x^+}|$ is the total number of violators. Summing over $\mathcal{V}_{q,x^+}$ in Equation 4 is equivalent to summing over $\mathcal{X}_q^-$ (as nonviolating x^- contribute zero to the loss), but allows us to drop the denominator $r_q^m(x^+)$ if we perform SGD on the loss and uniformly sample a violator in the sampling step (see (Weston et al., 2010) for details).

To optimize (1) with the WARP loss via SGD, we substitute $\mathcal{L}(r_q(x^+))$ in (1) with (4) and replace the sum with an expectation over $(q,\ x^+,\ x^-)$:

$$E[\mathcal{L}(r_q^m(x^+))[1 - f_q(x^+) + f_q(x^-)]_+ - \lambda f_q(x^+)] \qquad (5)$$

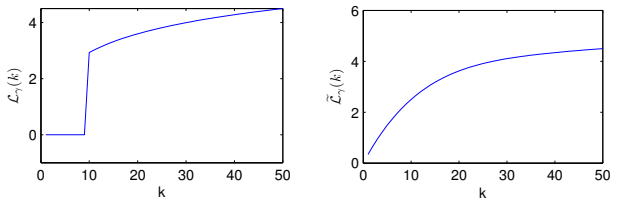

Figure 1. Example plots of $\mathcal{L}_\gamma(\cdot)$ and $\widetilde{\mathcal{L}}_\gamma(\cdot)$ for $|\mathcal{X}_q^-| = 50,\ \gamma = 10,\ \mathcal{L}(k) = \sum_{i=1}^{k} \alpha_i,\ \alpha_i = \frac{1}{i}$

2.2. Early stopping with the rank-γ truncated loss

Equation 5 can be optimized by repeatedly sampling a triplet $(q,\ x^+,\ x^- \in \mathcal{V}_{q,x^+})$ from the training set and performing gradient descent on the sample triplet loss. For a given (q, x^+) pair, computing $r_q^m(x^+)$ is expensive, as $f_q(x)$ has to be evaluated for each point in $x^+ \cup \mathcal{X}_q^-$. Thus, (Weston et al., 2010) proposed approximating $r_q^m(x^+)$ by $\lfloor |\mathcal{X}_q^-|/N_k \rfloor$, where N_k is the number of samples drawn with replacement from $\mathcal{X}_q^-$ until a *violator* is found. This scheme is equivalent to replacing $\mathcal{L}(r_q^m(x^+))$ in (5) with the expected loss $E_{N_k}[\mathcal{L}(\lfloor |\mathcal{X}_q^-|/N_k \rfloor)]$. Despite the use of this sampling scheme, the runtime of the overall algorithm can still be quite slow. Even at the initial stages before the metric is well trained, if we sample (q, x^+) such that x^+ is highly ranked, we may have to calculate $f_q(x)$ up to $|\mathcal{X}_q^-|$ times per iteration (since $\mathcal{L}(\lfloor |\mathcal{X}_q^-|/N_k \rfloor) = 0$ for $N_k > |\mathcal{X}_q^-|$), which can be computationally expensive for high-dimensional data. To speed up the process, we propose a simple modification: We stop sampling once $|\mathcal{X}_q^-|/\gamma$ samples have been drawn, where $\gamma \geq 1 \in \mathbb{Z}$ is a parameter. We now show how this "early stopping" is a natural consequence of a modified loss function.

Given an initial loss function $\mathcal{L}(\cdot)$, let us define the *rank-γ truncated loss* $\mathcal{L}_\gamma(\cdot)$:

$$\mathcal{L}_\gamma(k) = \begin{cases} 0 & k < \gamma \\ \mathcal{L}(k) & k \geq \gamma \end{cases}$$

Here, γ takes values in $1 \cdots |\mathcal{X}_q^-|$ and $\gamma = 1$ recovers $\mathcal{L}(k)$. If we replace $\mathcal{L}(\cdot)$ with the corresponding $\mathcal{L}_\gamma(\cdot)$, it is clear that since $\mathcal{L}_\gamma(\lfloor |\mathcal{X}_q^-|/N_k \rfloor) = 0$ for $N_k > |\mathcal{X}_q^-|/\gamma$, it is unnecessary to continue sampling after $|\mathcal{X}_q^-|/\gamma$ samples.

Although $\mathcal{L}_\gamma(\cdot)$ does not satisfy (3), we can verify[1] that the expected loss $\widetilde{\mathcal{L}}_\gamma(k) = E_{N_k}[\mathcal{L}_\gamma(\lfloor |\mathcal{X}_q^-|/N_k \rfloor)]$ does, however, satisfy (3), which is desirable for optimizing the top of the ranking[2]. This implies that $\widetilde{\mathcal{L}}_\gamma(k)$ is a convex ordered weighted averaging (OWA) operator in the sense of (Usunier et al., 2009). Figure 1 shows examples of $\mathcal{L}_\gamma(\cdot)$ and $\widetilde{\mathcal{L}}_\gamma(\cdot)$ for $|\mathcal{X}_q^-| = 50,\ \gamma = 10$. We can see that although $\mathcal{L}_\gamma(k)$ assigns zero loss for all $k < \gamma$, the expected loss strictly increases with k. This observation is consistent for all values of $|\mathcal{X}_q^-|$ and γ in our experiments, and we attempt to provide some intuition for this. In the sequel, a *trial* denotes drawing a single sample uniformly from $\mathcal{X}_q^-$, while a *run* denotes a complete process of sampling from $\mathcal{X}_q^-$ until either $N_k > |\mathcal{X}_q^-|/\gamma$ or a violator is found.

Consider a query q and two points $x_1^+, x_2^+ \in \mathcal{X}_q^+$, such that $r_q^m(x_1^+) < r_q^m(x_2^+) < \gamma$. In the deterministic setting where we calculate $r_q^m(\cdot)$ explicitly for each, $\mathcal{L}_\gamma(\cdot)$ would assign both x_1^+ and x_2^+ the same loss of 0. However, when approximating $r_q^m(x^+)$ stochastically, since $r_q^m(x_1^+) < r_q^m(x_2^+)$, x_2^+ has a higher probability of sampling a violator at each trial, and thus is likely to incounter violators earlier in the sampling process (i.e. stopping at smaller N_k) as compared to x_1^+. As $\mathcal{L}_\gamma(\lfloor |\mathcal{X}_q^-|/N_k \rfloor)$ is nonincreasing (in fact, usually decreasing) with N_k, this should lead to higher expected losses in general.

3. Optimization

With the modification in Section 2.2, the final loss we wish to minimize is

$$E[\widetilde{\mathcal{L}}_\gamma(r_q^m(x^+))[1 + f_q(x^+) - f_q(x^-)]_+ + \lambda f_q(x^+)] \quad (6)$$

Though we proposed the rank-γ truncated loss to justify early stopping, it is implicitly generated by stopping the sampling process early. Thus, no further modification to the algorithm is required.

We can optimize (6) via SGD by sampling (q, x^+, x^-, N_k) appropriately. Since we wish to optimize (6) over $S_{d,m}^+$, we could project W onto $S_{d,m}^+$ by performing a spectral decomposition at each step. However, this is computationally expensive for high dimensional data. It has been shown (Absil et al., 2008) that $S_{d,m}^+$ is an *embedded Riemannian manifold*, which is a smooth subset of the ambient space $\mathbb{R}^{d \times d}$. Thus, we can use recently proposed Riemannian optimization methods to learn W, which we briefly review below. The interested reader can find a complete treatment in (Absil et al., 2008).

3.1. Optimization on Riemannian Manifolds

We begin with the notions of *tangent space* and *retraction*. Each point W in an embedded manifold $\mathcal{M}$ has a *tangent space* denoted as $\mathcal{T}_W\mathcal{M}$, which is the set of tangents to smooth curves within $\mathcal{M}$ passing through W. A *retraction* is any function $R : \mathcal{T}_W\mathcal{M} \to \mathcal{M}$ that satifies

[1] Given a fixed $L(k) = \sum_i^n \alpha_i$, the expectation $\widetilde{\mathcal{L}}_\gamma(k)$ can be explicitly evaluated for $k = 1 \cdots |\mathcal{X}_q^-|$. This fully defines $\beta_1 = \widetilde{\mathcal{L}}_\gamma(1), \beta_2 = \widetilde{\mathcal{L}}_\gamma(2) - \widetilde{\mathcal{L}}_\gamma(1)$ etc. which can be verified to fufil the conditions.

[2] It can be shown that requiring $\mathcal{L}_\gamma(\lfloor |\mathcal{X}_q^-|/N_k \rfloor)$ to be nonincreasing for $N_k = 1 \cdots |\mathcal{X}_q^-|$ is a sufficient condition for $\widetilde{\mathcal{L}}_\gamma(\cdot)$ to satisfy (3) for *any* $\mathcal{L}(\cdot)$ that satisfies (3) and for any value of γ. We will include the proof in the extended version of the paper.

the properties of *centering* and *local rigidity* (Absil et al., 2008). The mathematically ideal retraction is called the *exponential map*, which is usually computationally expensive. Instead, one can use retractions which approximate the exponential map, and still retain the local convergence properties of the exponential map.

A function $f(W)$ defined over an embedded Riemannian manifold $\mathcal{M}$ can be optimized via gradient descent. Given a current candidate solution W_t and a retraction R_W, we need to perform 2 steps at each iteration:

1) Calculate $W^+ = W_t - \eta\tilde{\nabla} f(W_t)$, where η is the step size and $\tilde{\nabla} f(W_t)$ is the *Riemannian gradient* at W_t.
2) Map W^+ back to $\mathcal{M}$: $W_{t+1} = R_W(W^+)$.

Given an embedded manifold $\mathcal{M}$ and a function f defined in the ambient space, the Riemannian gradient of f at W is simply the orthogonal projection of the standard Euclidean gradient $\nabla f(W)$ onto $\mathcal{T}_W\mathcal{M}$. For $S^+_{d,m}$, this projection is given by the following lemma.

Lemma 3.1 *Given a point $W = YY^\mathsf{T} \in S^+_{d,m}$, the orthogonal projection of a matrix Z in the ambient space $\mathbb{R}^{d\times d}$ onto $\mathcal{T}_W S^+_{d,m}$, is given by $P_{T_W}(Z) = \xi$, where*

$$\xi = \xi^s + \xi^p; \quad \xi^s = P_y \frac{Z + Z^\mathsf{T}}{2} P_y, \qquad (7)$$
$$\xi^p = P_y^\perp \frac{Z + Z^\mathsf{T}}{2} P_y + P_y \frac{Z + Z^\mathsf{T}}{2} P_y^\perp$$

and $P_y = YY^\dagger$, $P_y^\perp = I - P_y$

Proof See Proposition 5.2 in (Vandereycken & Vandewalle, 2010)

The retraction we require is given in the following lemma:

Lemma 3.2 *Let $W \in S^+_{d,m}$ and $\xi,\ \xi^p,\ \xi^s$ be as defined in Lemma 3.1. Then, the function $\mathcal{R}_W(\xi) = VW^\dagger V$ where*

$$V = W + \frac{1}{2}\xi^s + \xi^p - \frac{1}{8}\xi^s W^\dagger \xi^s - \frac{1}{2}\xi^p W^\dagger \xi^s$$

is a second-order retraction from the tangent space $\mathcal{T}_W S^+_{d,m}$ to $S^+_{d,m}$.

Proof See Proposition 5.10 in (Vandereycken & Vandewalle, 2010)

We can now perform Riemannian stochastic gradient descent to minimize Equation (6) over $S^+_{d,m}$. Given a current estimate W_t, this can be done as follows:

1: Sample a pair $(q, x^+ \in \mathcal{X}_q^+)$; sample $(N_k, x^- \mid q, x^+)$.
2: Calculate $\xi^s,\ \xi^p$ as in (7) for
$Z = -\eta\nabla\Big[\mathcal{L}\Big(\Big\lfloor \frac{|\mathcal{X}_q^-|}{N_k} \Big\rfloor\Big)[1 - f_q(x^+) + f_q(x^-)]_+ + \lambda f_q(x^+)\Big]\Big|_{W_t}$
3: $W_{t+1} \leftarrow \mathcal{R}_W(\xi)$

Algorithm 1 Symmetric_gradient_update

Input: Initial matrix $L \in \mathbb{R}^{d\times m}$ such that $W = LL^\mathsf{T}$, $U,\ V$ such that $-\eta\nabla_W = UV^\mathsf{T}$
Output: M such that $MM^\mathsf{T} = \mathcal{R}_W(P_{T_W}(W - \eta\nabla W))$
$L^\dagger = (L^\mathsf{T}L)^{-1}L^\mathsf{T}$
$A_1 = L^\dagger U;\ A_2 = L^\dagger V;\ S = A_1^\mathsf{T}A_2;\ \hat{A}_1 = LA_1$
return $M = L + (U - \frac{1}{2}\hat{A}_1 + (\frac{3}{8}\hat{A}_1 - \frac{1}{2}U)S)A_2^\mathsf{T}$

Algorithm 2 FRML-WARP

Input: $L \in \mathbb{R}^{d\times m}$ such that $LL^\mathsf{T} = W$, data matrix $\mathcal{X} \in \mathbb{R}^{d\times n}$, relevant/irrelevant sets $\mathcal{X}_q^+/\mathcal{X}_q^-\ \forall q \in \mathcal{X}$, sampling threshold γ
repeat
  Draw q from $\mathcal{X}$; Draw x_j from $\mathcal{X}_q^+$; $N_k \leftarrow 0$
  repeat
    Sample x_l from $\mathcal{X}_q^-$
  until $N_k > |X_i^-|/\gamma$ or $f_q(x^-) - f_q(x^+) > 1$
  $\hat{r}_1 = \lfloor |\mathcal{X}_q^-|/N_k \rfloor;\ \vec{v}_{qj} \leftarrow q - x_j;\ \vec{v}_{ql} \leftarrow q - x_l$
  if $f_q(x^-) - f_q(x^+) > 1$ **then**
    $c_{qj} = -\eta(\mathcal{L}(\hat{r}_1) + \lambda),\ c_{ql} = \eta(\mathcal{L}(\hat{r}_1))$
  else
    $c_{qj} = -\eta(\lambda),\ c_{ql} = 0$
  end if
  $U = [c_{qj}\vec{v}_{qj},\ c_{ql}\vec{v}_{ql}];\ V = [\vec{v}_{qj},\ \vec{v}_{ql}]$
  $L \leftarrow$ Symmetric_gradient_update (L, U, V)
until max iterations exceeded or validation error does not improve
return $W = LL^\mathsf{T}$

Steps 2 and 3 can be combined into a single function, which is given by Algorithm 1. With the choice of $\Omega_L(W)$ as a regularizer, the sample gradient Z is a symmetric matrix, which simplfies the derivation of the update step. The update is similar in spirit to the one in (Shalit et al., 2012), but unlike their case where the gradient is nonsymmetric, the quadratic cross terms cancel in our case, leading to a simpler update. The complete derivation is given in the supplementary material. Our complete approach, FRML-WARP, is given by Algorithm 2.

3.2. Computational Complexity

In this section, we analyze the computational complexity of Algorithm 2. In the sequel, let d be the input dimensionality, m, the rank of W, and r, the rank of the gradient UV^T. We also consider the minibatch approach where steps 2-11 of Algorithm 2 are repeated b times, and UV^T is the averaged gradient over b examples. When $b = 1$, $\mathrm{rank}(UV^\mathsf{T})$ is two when a violator is found in steps 3-5 and one otherwise (since $c_{ql} = 0$, we can discard the corresponding column of U). For simplicity, we assume that a minibatch of size b is of rank $2b$ (when $b << d$).

The runtime of Algorithm 2 depends on two factors: The runtime of the sampling process (steps 3-5) and the runtime of Algorithm 1. Every sampling run can require up to $|\mathcal{X}_i^-|/\gamma$ checks for a violator, each of which is $O(dm)$. Like the WSABIE algorithm (Weston et al., 2010), Algorithm 2 is best suited for challenging problems where only a few relevant examples end up at the top of the ranking (since, in expectation, many more $O(dm)$ checks are required in step 4 as x_j approaches the top of the ranking).

The runtime of Algorithm 1 is $O(d \cdot \max(r, m)^2)$. The two potentially expensive steps are computing the pseudoinverse of L, which is $O(dm^2)$, and the $O(dr^2)$ multiplication involving S in step 3. In practice, one can avoid ever computing $L^\dagger$ explicitly; A_1 (and A_2, by symmetry) can be obtained by solving $(L^\mathsf{T}L)A_1 = L^\mathsf{T}U$ via Cholesky decomposition which is faster than first computing $L^\dagger$. Another approach would be to use a rank-one update proposed in (Shalit et al., 2012). Despite the fact that this is $O(dm)$ (vs $O(dm^2)$), it can be computationally expensive in practice when $b > 1$ as, instead of computing a single rank $2b$ outer product in step 3 of Algorithm 1, $2b$ rank-one outer products need to be computed as $L^\dagger$ must be updated incrementally. Empirically we found that obtaining A_1 directly was generally faster than the rank-one update in our experiments, where we set $b = 5$.

The choice of Ω_L (Equation 2) as a regularizer is now clear as 1) it gives rise to symmetric gradients and 2) it decomposes over (q, x^+), giving rise to a rank-one sample gradient matrix. Other choices of $\Omega(W)$, such as the graph Laplacian (Hoi et al., 2008) or the Frobenius norm generally give rise to high-rank gradients and do not admit such a decomposition in general. Naively using the full-rank gradient at each time step would render our algorithm unacceptably slow as the complexity of Algorithm 2 is quadratic in r. In this case, it may be faster to work with the unfactored form of the gradient directly. Alternatively, since every matrix $\Omega(W)$ can be expressed as $\sum_{i=1}^{k} u_i v_i^\mathsf{T}$ where k is the rank of $\Omega(W)$ and $u_i,\ v_i$ are vectors, we can obtain an unbiased estimate of the full gradient at each stochastic gradient step as $u_i v_i^\mathsf{T}$, where i is sampled uniformly at random from $1 \cdots k$. This extension is left for future work.

4. Related Work

Distance metric learning is a well studied problem, of which representative methods are Information-Theoretic Metric Learning (ITML) (Davis et al., 2007), Large Margin Nearest Neighbor (LMNN) (Weinberger & Saul, 2009) and the method of (Xing et al., 2003). Our work is most strongly inspired by Metric Learning to Rank (McFee & Lanckriet, 2010), which introduced the notion of optimizing a Mahalanobis metric for a ranking loss. A comprehensive survey of other metric learning techniques can be found in (Bellet et al., 2013).

A variety of methods have been proposed to circumvent the $O(d^3)$ decomposition step usually required to enforce $W \succeq 0$. (Torresani & Lee, 2007) considered restricting W to be low-rank by substituting $W = L^\mathsf{T}L$, $L \in \mathbb{R}^{m \times d}$ and optimizing the loss function with respect to L. This method, while similar to ours, does not account for the invariance of L to orthonormal transformations, which results in non-isolated minimizers unlike our approach. As a result, we found empirically that this method exhibits poorer convergence and sensitivity to step size compared to our proposed method. Other methods such as (Shen et al., 2009) and (Ying & Li, 2012) require to find the largest eigenvalue of W at each iteration, which scales as $O(d^2)$. Hence, they do not scale to high-dimensional data.

WSABIE (Weston et al., 2010), jointly learns low-rank embeddings of training points and labels, whereas our method learns a single embedding over points for retrieval. OASIS (Chechik et al., 2009) is also similar to our method, but does not enforce low rank or positive definiteness of the metric, and optimizes for AUC.

Most similar to our approach is the PSD-1 variant of LORETA (Shalit et al., 2012), with a few key differences: Instead of an inner-product based similarity, we consider a distance-based similarity. This is more suitable for low-dimensional visualizations of the data, and also for incorporating ideas from semi-supervised learning such as the graph Laplacian regularizer, which cannot be directly applied to the inner-product case. Furthermore, we incorporate the WARP loss to improve performance at the top of the rankings, and we exploit the symmetry of the sample gradients for a more elegant update (Algorithm 1).

Recently, methods have been proposed to avoid scoring the full training set for minimizing listwise losses such as MAP (Shi et al., 2012) for the recommendation setting. It would be interesting to see if these methods could be adapted to the distance metric learning task as well.

4.1. Connection to Large Margin Nearest Neighbor

FRML can be shown to be equivalent to LMNN if in (1), we remove the rank constraint; set $\Omega(W) = \Omega_L(W)$; set $\mathcal{X}_q^+/\mathcal{X}_q^-$ to be the target neighbors/impostors of q; set $\mathcal{L}(\cdot)$ to be the identity; and replace $r_q(x^+)$ with the convex upper bound

$$\widetilde{r_q}(x^+) = \sum_{x^- \in \mathcal{X}_q^-} [1 - f_q(x^+) + f_q(x^-)]_+$$

Recent work (Do et al., 2012) has shown that LMNN can be considered to be jointly optimizing multiple SVM subproblems, with a parameter vector where certain entries are dependent on each other. By casting LMNN in our frame-

work, we show that LMNN can be viewed as a learning-to-rank problem which optimizes the mean AUC loss on all training examples (with $\mathcal{X}_q^+$ comprising *only* target neighbors for each query) over a shared parameter matrix.

5. Experiments

To evaluate our proposed method, we conducted two sets of experiments. In the first experiment, we evaluated the retrieval performance and training time of various metric learning algorithms on three high-dimensional datasets: ImageNet (Deng et al., 2009), CAL10K (Tingle et al., 2010) and MagnaTagatune (Law et al., 2009). In the second experiment, we compared the performance of our method with competing algorithms on a subset of the `covertype` dataset with a large number of training examples relative to data dimensionality.

5.1. High-dimensional datasets

For all experiments in this section, we compare Metric Learning to Rank (MLR) (McFee & Lanckriet, 2010), OASIS (Chechik et al., 2009), LORETA (Shalit et al., 2012), FRML-WARP, and FRML-AUC, which optimizes Equation 5 with $\mathcal{L}(\cdot)$ set to the identity. For MLR, we use the MLR-ADMM implementation (Lim et al., 2013) and report seperately the cases where the metric was optimized via the AUC (Joachims, 2005) and MAP (Yue et al., 2007) separation oracles, denoted as MLR-AUC and MLR-MAP respectively. For OASIS, we used the nonsymmetric variant while for LORETA, we used the PSD-1 variant. For LORETA and both variants of FRML, we report performance for $m \in \{20, 30, 50, 100, 200\}$, where m is the rank of the learned metric. Additionally, for FRML-WARP, we varied γ in $\{1, 10, 25\}$.

Given a query q and learned metrics W, a predicted ranking was induced on $\mathcal{X}$ by sorting $q^\mathsf{T} W x$ for $x \in \mathcal{X}$ in decreasing order for similarity-based methods (LORETA, OASIS), while for distance-based methods, the predicted ranking was induced by sorting $\|q - x\|_W$ for $x \in \mathcal{X}$ in increasing order. For each experiment, we report AUC, MAP and precision-at-k (P@k) of these predicted rankings.

5.1.1. ImageNet Retrieval

For this experiment, 100 images were chosen from each of 20 categories from the ImageNet repository, and each image was represented using 1000-dimensional SIFT codeword histograms obtained from the ImageNet database. Supervision was provided at the class level for ranking-based algorithms: For each training point $q \in \mathcal{X}$, $\mathcal{X}_i^+$ was defined as the set of all same-class images to the query, and $\mathcal{X}_i^-$ the set of all different-class images to the query.

We additionally provided comparisons with Information-Theoretic Metric Learning (ITML) (Davis et al., 2007) and Large Margin Nearest Neighbor (LMNN) (Weinberger & Saul, 2009), as they can work with class membership labels. For LMNN, we optimized over the (full-rank) factored matrix L instead of W, as directly optimizing W was not computationally feasible. We also tried running the dimensionality reduction variant of LMNN using the code from (Weinberger, 2014), but did not obtain competititive performance, thus we do not report the results.

For ITML, the slack parameter γ was varied over $\{1, 10, \ldots, 10^6\}$ For LMNN, the push-pull parameter μ was varied over $\{0.1, 0.2, \ldots, 0.9\}$ and the number of target neighbors was fixed to 10. For MLR and OASIS, C was varied over $\{1, 10, \ldots, 10^6\}$. For each method, the hyperparameters with the best MAP performance on the validation set were selected.

For LORETA and FRML, the step size η was chosen by MAP performance on a held-out set every 100000 iterations. For a given setting of m, W was initialized as the product LL^T, where the entries of L were generated by the standard normal distribution. For both variants of FRML, the trade-off parameter λ was fixed at 0.1 and we used a minibatch of size 5. Each of the online methods were run until 300,000 training triplets were observed.

5.1.2. CAL10K

We used a subset of the CAL10K dataset, which was provided as ten 40/30/30 splits of a collection of 5419 songs. We followed the approach of (McFee et al., 2012) to process the audio data. Five-second sequences of MFCC vectors were first drawn from a set of held-out songs. These sequences were then collected into bags of features, randomly permuted and then clustered to form a codebook of size 2048. Each song was then represented as a vector quantization histogram over this codebook.

For each song q, $\mathcal{X}_q^+$ was defined as the subset of songs in the training set performed by the top 10 most similar artists to the performer of q, where similarity between artists was measured by the number of shared users in a sample of collaborative filter data from last.FM. Due to the non-transitive nature of the similarity in each case, we measured performance only with the ranking-based methods: MLR, LORETA, OASIS and both FRML variants.

5.1.3. MagnaTagatune

The Magnatagatune dataset comprises 25,860 30-second audio clips, each of which has been annotated by humans via the TagATune game. Each clip is assigned a corresponding 188-dimensional binary tag vector, where a 1 in a given position indicates that a tag applies to the song. In our experiment, we only worked with songs with at least

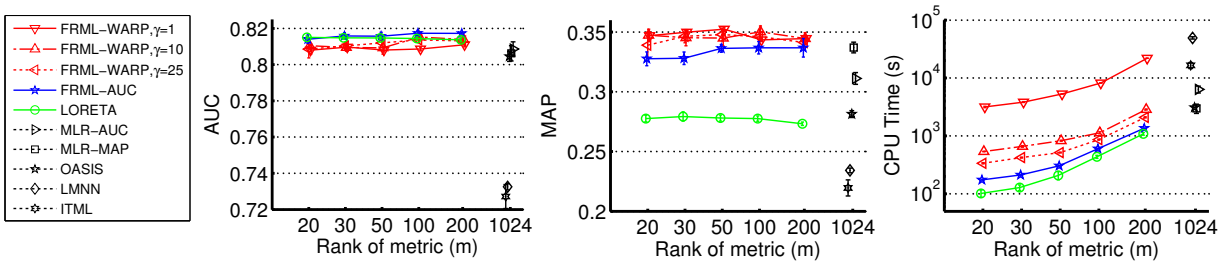

Figure 2. Performance of various algorithms on the ImageNet dataset (best viewed in color). Curves indicate mean performance over 5 folds. Error bars indicate standard error of the mean.

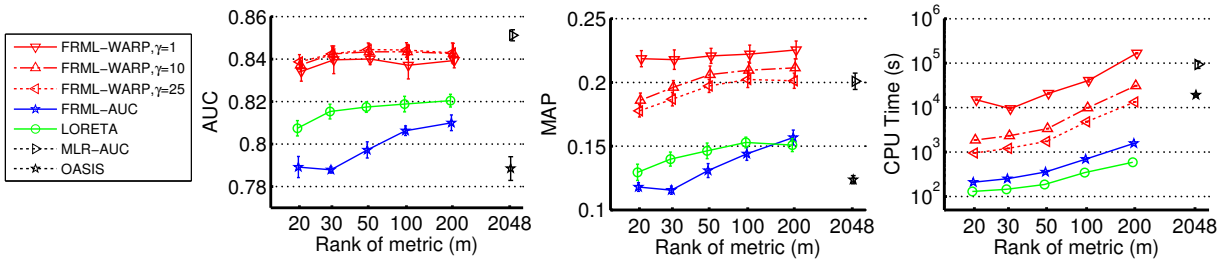

Figure 3. Performance of various algorithms on the CAL10K dataset (best viewed in color). Curves indicate mean performance over 5 folds. Error bars indicate standard error of the mean.

5 annotations, giving us 10,716 songs in total. These were split into 4 folds in a 75/25 train/test split.

To obtain an audio feature representation, we follow the method of (Su et al., 2014). Spectrogram extraction was first performed on the raw audio for a series of frames, followed by feature aggregation. Each bag of feature vectors was then encoded via sparse coding using a pre-trained dictionary of size 1024, after which pooling and power normalization were performed to obtain a single vector representation for each clip.

Given a song q, we set $\mathcal{X}_q^+$ to be the top 5% of songs in the training set that were most similar to q, where similarity was measured by the cosine similarity: $\text{sim}(q, x) = \frac{q^T x}{|q||x|}$. $\mathcal{X}_q^-$ was defined to be the subset of songs in the training set which do not share any tags in common with q (*i.e.* having a cosine similarity of 0). As in CAL10K, only the ranking-based methods were compared.

5.1.4. Results

Figure 2 shows the performance of the various algorithms on the ImageNet retrieval task. On the MAP metric, FRML-WARP is able to match or outperform existing algorithms even when $m = 20$. Setting $\gamma = 25$ did not significantly impact performance, while reducing training time by an order of magnitude. Both classification-based algorithms did not perform as well as the ranking-based algorithms on either retrieval metric.

Figure 3 shows the performance of the various algorithms on the CAL10K retrieval task. On the MAP metric, FRML-WARP ($\gamma = 1$) outperforms competing algorithms across all values of m, while FRML-WARP ($\gamma = 25$) still outperforms or matches other methods for $m \geq 50$. On both AUC and MAP metrics, FRML-WARP performance degrades modestly as γ increases. We did not report MLR-MAP performance as it did not converge in a reasonable amount of time (10^6 seconds).

Figure 4 shows the performance of the various algorithms on the Magnatagatune retrieval task. Here, as in the ImageNet experiment, setting $\gamma = 25$ offers a $\sim 10\times$ reduction in the FRML-WARP training time over the $\gamma = 1$ case, without suffering any appreciable loss in performance on either metric. FRML-WARP outperforms all other algorithms across both metrics, for $m \geq 50$. In this experiment, MLR-MAP also failed to converge within 10^6 seconds.

Table 1 reports the precision-at-k (P@k) performance for $k = \{1, 10\}$ on all three datasets. For the low-rank methods, we reported results for the value of m which had the best MAP performance on the validation set. FRML-WARP has the best precision-at-k performance across all datasets, showing the effectiveness of the WARP loss at optimizing the top of the ranking. We also observe that LMNN seems to perform poorly on AUC and MAP, but still does relatively well on P@k. This is probably because LMNN focuses only on optimizing 10 target neighbors, which lower its performance on AUC and MAP which consider *recall* performance (unlike P@k). This was also observed in our second experiment.

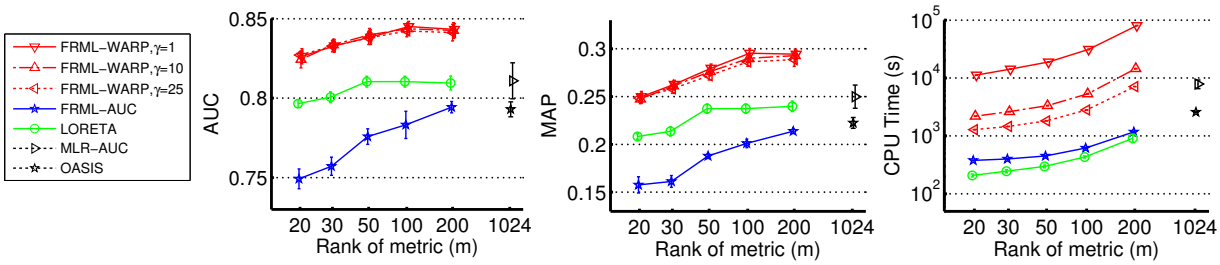

Figure 4. Performance of various algorithms on the Magnatagatune dataset (best viewed in color). Curves indicate mean performance over 4 folds. Error bars indicate standard error of the mean.

Table 1. Precision-at-k performance on the three datasets. For FRML and LORETA, we report results for the value of m with the best MAP performance on the validation set. Bold numbers indicate the methods with the best performance.

		FRML-AUC	FRML-WARP			LORETA	OASIS	MLR-AUC	MLR-MAP	LMNN	ITML
			γ=1	γ=10	γ=25						
ImageNet	P@1	0.366	0.377	**0.380**	0.378	0.339	0.271	0.354	0.375	0.355	0.292
	P@10	0.344	0.357	**0.359**	0.356	0.324	0.280	0.330	0.357	0.320	0.264
CAL10K	P@1	0.262	**0.328**	0.297	0.292	0.221	0.182	0.271			
	P@10	0.226	**0.294**	0.262	0.256	0.193	0.160	0.241			
MagnaTagatune	P@1	0.470	0.521	**0.524**	0.519	0.444	0.460	0.467			
	P@10	0.414	0.485	**0.489**	0.480	0.412	0.408	0.425			

We observed also that the geometry of the data plays a role in MAP performance. Both FRML-AUC and LORETA are very similar with the main difference between how the ranking scores are modeled (Mahalanobis distance vs inner product), yet FRML-AUC outperforms LORETA on ImageNet while the converse is true on Magnatagatune. Thus, it is still an open question as to which modeling method is more suitable for a given dataset.

Even though the experimental results seem to indicate that our methods are 10-100$\times$ slower than LORETA or FRML-AUC, the reported time is the time needed to process 300,000 training triplets. However, we observed that given a *fixed* training time budget, FRML-WARP with a suitable setting of γ can achieve better MAP performance than either AUC method on the test set.

5.2. Low-dimensional dataset

In this experiment, we wish to evaluate the various algorithms on a low-dimensional, large dataset where the natural advantage of low-rank methods over full-rank methods (in terms of having fewer parameters to estimate reliably) is much less pronounced. We used a subset of the `covertype` dataset from the UCI repository, which comprises data from 7 classes. We sampled 10000 54-dimensional data points for our experiment, which we split into five 80/20 folds. For FRML and LORETA we fixed m = 30 and $\gamma = 1$, and followed the protocol of Section 5.1 for the other methods.

Table 2 reports the performance of various methods on this dataset. Again, MLR-MAP did not converge within 10^6 seconds so we did not report results. FRML-WARP outperforms other methods on top-of-the-ranking measures (MAP and P@k), indicating its suitability even on low-dimensional datasets.

Table 2. Performance of various methods on `covertype`. Bold numbers indicate the methods with the best performance.

	AUC	MAP	P@1	P@10
FRML-WARP	0.843	**0.511**	**0.811**	**0.689**
FRML-AUC	**0.852**	0.477	0.749	0.648
LORETA	0.823	0.427	0.804	0.671
MLR-AUC	0.851	0.481	0.762	0.665
OASIS	0.792	0.417	0.792	0.674
LMNN	0.661	0.307	0.782	0.666
ITML	0.835	0.458	0.762	0.663

6. Conclusion

We proposed an novel distance metric learning algorithm for ranking, and derived an efficient learning algorithm as well as a truncated sampling scheme for greater computational efficiency. Our experiments demonstrate that our proposed method outperforms existing methods on top-heavy ranking metrics while having substantially reduced computation time.

Acknowledgements

The authors acknowledge support from from Yahoo!, Inc., the Sloan Foundation, KETI under the PHTM program, and NSF Grants CCF-0830535 and IIS-1054960. Daryl Lim was supported by a fellowship from the Agency for Science, Technology and Research (A*STAR), Singapore.

References

Absil, Pierre-Antoine, Mahony, Robert E., and Sepulchre, Rodolphe. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

Bellet, Aurélien, Habrard, Amaury, and Sebban, Marc. A survey on metric learning for feature vectors and structured data. *CoRR*, abs/1306.6709, 2013.

Chechik, Gal, Sharma, Varun, Shalit, Uri, and Bengio, Samy. Large scale online learning of image similarity through ranking. In *IbPRIA*, pp. 11–14, 2009.

Davis, Jason V., Kulis, Brian, Jain, Prateek, Sra, Suvrit, and Dhillon, Inderjit S. Information-theoretic metric learning. In *International Conference on Machine Learning (ICML)*, 2007.

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-jia, Li, Kai, and Li, Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE CVPR*, 2009.

Do, Huyen, Kalousis, Alexandros, Wang, Jun, and Woznica, Adam. A metric learning perspective of svm: on the relation of lmnn and svm. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

Hoi, Steven C. H., Liu, Wei, and Chang, Shih-Fu. Semi-supervised distance metric learning for collaborative image retrieval. In *Proc. IEEE CVPR*, 2008.

Joachims, T. A support vector method for multivariate performance measures. In *International Conference on Machine Learning (ICML)*, 2005.

Law, Edith, West, Kris, Mandel, Michael I., Bay, Mert, and Downie, J. Stephen. Evaluation of algorithms using games: The case of music tagging. In *Proc. ISMIR*, 2009.

Lim, Daryl, Lanckriet, Gert R. G., and McFee, Brian. Robust structural metric learning. In *International Conference on Machine Learning (ICML)*, 2013.

McFee, Brian and Lanckriet, G.R.G. Metric learning to rank. In *International Conference on Machine Learning (ICML)*, 2010.

McFee, Brian, Barrington, Luke, and Lanckriet, Gert R. G. Learning content similarity for music recommendation. *IEEE Transactions on Audio, Speech & Language Processing*, 20(8):2207–2218, 2012.

Shalit, Uri, Weinshall, Daphna, and Chechik, Gal. Online learning in the embedded manifold of low-rank matrices. *Journal of Machine Learning Research*, 13:429–458, 2012.

Shen, Chunhua, Kim, Junae, Wang, Lei, and van den Hengel, Anton. Positive semidefinite metric learning with boosting. In *Advances in Neural Information Processing Systems 22*. 2009.

Shi, Yue, Karatzoglou, Alexandros, Baltrunas, Linas, Larson, Martha, Hanjalic, Alan, and Oliver, Nuria. Tfmap: optimizing map for top-n context-aware recommendation. In *Proc. ACM SIGIR*, 2012.

Su, L., Yeh, C.-C. M., Liu, J.-Y., Wang, J.-C., and Yang, Y.-H. A systematic evaluation of the bag-of-frames representation for music information retrieval. *IEEE Trans. Multimedia*, 2014.

Tingle, D., Kim, Y., and Turnbull, D. Exploring automatic music annotation with "acoustically-objective" tags. In *IEEE International Conference on Multimedia Information Retrieval*, 2010.

Torresani, Lorenzo and Lee, Kuang C. Large Margin Component Analysis. In *Advances in Neural Information Processing Systems*. 2007.

Usunier, Nicolas, Buffoni, David, and Gallinari, Patrick. Ranking with ordered weighted pairwise classification. In *International Conference on Machine Learning (ICML)*, 2009.

Vandereycken, Bart and Vandewalle, Stefan. A riemannian optimization approach for computing low-rank solutions of lyapunov equations. *SIAM J. Matrix Analysis Applications*, 31(5):2553–2579, 2010.

Weinberger, Kilian Q. and Saul, Lawrence K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10: 207–244, June 2009.

Weinberger, Killian Q. LMNN 2.4 code, 2014. URL http://www.cse.wustl.edu/~kilian/code/files/mLMNN2.4.zip.

Weston, Jason, Bengio, Samy, and Usunier, Nicolas. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, 81:21–35, 2010.

Xing, Eric P., Ng, Andrew Y., Jordan, Michael I., and Russell, Stuart. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, 2003.

Ying, Yiming and Li, Peng. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13:1–26, January 2012.

Yue, Yisong, Finley, Thomas, Radlinski, Filip, and Joachims, Thorsten. A support vector method for optimizing average precision. In *Proc. ACM SIGIR*, 2007.