Adversarial Learning-to-Rank Based on Variational Divergence Minimization

Inagaki Ryo

Graduate School of Library, Information and Media Studies

University of Tsukuba

March 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recently, research on ranking has been actively conducted in information retrieval. In this thesis, we focus on the field of document retrieval. In document retrieval, ranking was traditionally performed using a generative model [1, 2]. This approach is based on a language model and ranks documents by the likelihood of the query according to the language model. However, one of the shortcomings of this model is that it is difficult to combine and handle many features between query and document. To overcome this problem with the strength of machine learning, ranking model has shifted to a discriminative model represented by learning-to-rank [3].

Learning-to-rank uses machine learning technologies to train the ranking model and has shown great value in many fields, such as document retrieval, recommender system and question answering. In the document retrieval, the learning-to-rank method learns an optimal way of combining features (such as TF-IDF [4] and BM25 [5]) extracted from query–document pairs through discriminative training. Given a query, the ranking model assigns a score to each document, and ranks the documents in descending order of the scores. The score represents the relevance of documents with respect to a query. By combining learning-to-rank with rich learning methods (such as deep neural networks and decision tree), it has become possible to realize a flexible and powerful ranking model. However, it has a problem that it cannot find useful features from a large amount of unlabeled data. Therefore, in recent years, studies have been conducted to try to solve this problem by learning generative model and discriminative model at the same time.

In recent years, research on applying adversarial learning to learning-to-rank has been actively conducted. Adversarial training is a learning method proposed by Generative Adversarial Nets [6] where two models (Generator, Discriminator) improve themselves by beating the other one at every round of this competition. The generator makes its own generation probability distribution close to the train data distribution and maximizes the probability that the discriminator makes a mistake. On the other hand, the discriminator attempts to distinguish whether the sample was obtained from ground-truth or the generator. This machine learning framework has been successfully applied to various fields such as image generation, natural language processing [7, 8] and speech processing [9, 10]. In particular, there is a lot of research on its application in image generation technology, and it is now possible to generate images that can even be mistaken as real images. IRGAN [11] was the first attempt to propose a minimax game framework to train information retrieval systems. They used the generator and discriminator to solve the problem where learning to rank has difficulty in dealing with unlabeled data. The generator aims to generate (or select) documents that look like the ground-truth relevant documents, which may fool the discriminator. On the other hand, the discriminator aims to make a clear distinction between the ground-truth relevant documents and the ones generated by its opponent generator. By repeating the generator and discriminator training, the generator will be able to sample more relevant documents, and the discriminator will be able to identify documents more accurately. Adversarial training for Information Retrieval (AdvIR) [12] also applies adversarial training to the training of the traditional ad-hoc information retrieval model. In addition to adversarial sampling, they generate documents that are difficult to judge by adding adversarial perturbation on top of the existing documents. In these approaches, GAN-divergence is used to minimize the difference between the probability distribution of the generator and the probability distribution of the training data, but it is not known whether it is optimal for document retrieval. In the field of image generation, there are examples where PC-divergence has been used to generate higher resolution images than GAN-divergence [13], and where high holdout likelihood has been achieved using KL-divergence compared to GAN-divergence [14]. In addition, the difficulty of stable learning has been reported [15, 12] as a problem of IRGAN.

The aforementioned question motivates us to approach a new ranking model with adversarial training. In the original GAN paper, the authors show that it is possible to train a generative model by approximate minimization of the GAN-divergence. It is a special case of divergence estimation. f-GAN [14] generalized it to arbitrary f-divergences and proposed variational divergence minimization framework for image generation. Therefore, in this thesis, in order to investigate the effects of different f-divergence for document retrieval, we propose a ranking model that uses f-divergence [16] minimization approach on adversarial learning-to-rank. We call this model IRf-GAN. It is possible to estimate the difference between the probability distribution of the generator and the training data due to various divergences by using f-divergence. The smaller the f-divergence, the more the generator will be able to sample relevant documents. In addition, for a stable learning, we modified training algorithm which took a single gradient update on both the generator and discriminator for each iteration step, and in the pairwise method, we used weighted sampling and sampling based on the probability obtained from the Bradley-Terry model. The purpose of this thesis is thus to investigate the effects of different f-divergence functions for document retrieval and make the training more stable by modifying the training algorithm to perform single-step gradient update for both the generator and discriminator at each iteration step and improving a sampling method for ground-truth and fake documents in the generator.

1.2 Contributions

The main contributions of this thesis are summarized as follows:

1. We proposed a novel adversarial training framework that applies f-divergence minimization to the training of the generator and discriminator that allows us to investigate the effect of different f-divergence functions on adversarial learning-to-rank, and to achieve stable learning with the single-step gradient method to update generator

and discriminator parameters in the same iteration and pair sampling method using weighting and pair comparison models.

2. To understand the effectiveness of our approach, we conduct a series of experiments using MSLR-WEB30K, Yahoo Learning to Rank Challenge corpus, MQ2008 and MQ2008-semi. Experiments on any dataset demonstrate that proposed framework for adversarial learning-to-rank shows better performance than the baseline approach based on either pointwise or pairwise model, and it is suggested that the number of features in the dataset used for learning affect the identification of optimal divergence in adversarial learning-to-rank.

1.3 Outline

This thesis consists of six chapters. In chapter 2, we describe the related works. First, we describe traditional information retrieval models (such as Boolean Model, Vector Space Model and Language Model), and then introduce the learning-to-rank and adversarial training, and finally, we introduce the combination of these two techniques that are at the core of this research. In chapter 3, we describe the proposed method. We take a closer look at the objective function of the proposed method and how to learn the generator and discriminator of pointwise and pairwise method, respectively. In chapter 4, we introduce the setup of the experiment, including the dataset, evaluation metrics, and baseline, and in chapter 5, we analyze and discuss the results of the experiment. Finally, in chapter 6, we present the conclusion to this work and propose directions for future work.

Chapter 2

Related Work

In this chapter, we describe related work in my research. First, we describe the Boolean Model, Vector Space Model, and Language Model as traditional generative models in information retrieval. Then, we explain learning to rank as a discriminative model, and explain the research on adversarial training, which is the key in this research. Finally, we introduce recent research combining learning to rank and adversarial training.

2.1 Information Retrieval

Information retrieval is to retrieve relevant documents from a collection of information system resources by various information retrieval strategies, thereby allowing users to easily access the information that are satisfying a user's information need [17]. In order to effectively search for relevant documents by those strategies, it is common to transform the documents into an appropriate representation, and the traditional information retrieval model differs in the way the documents are represented.

2.1.1 Boolean Model

The Boolean Model [18] is the earliest model that treats a document as a collection of words. When a query is given by a user, the document containing the word that matches the query is regarded as the related document. This model uses operators such as "AND, OR, NOT" to match documents with logical expressions based on Boolean algebra as queries. For example, "X AND Y" means that the document must contain two words, X and Y. "X OR Y" means that it must contain at least one of the words X and Y, and "NOT X" means that the document must not contain the word X. Such search method is used in libraries' online catalogs, such as OPAC and search engines, which has the advantage of being fast to search and easy to understand, but have the disadvantage of not being able to rank the output documents.

2.1.2 Vector Space Model

In the case of Vector Space Model [19], queries and documents are represented as vectors, and the vectors for each term are weighted according to their relative importance. TF-IDF [4] is often used to calculate this weight. It is the product of TF, which indicates the number of occurrences of the word in the document, and IDF, which is the inverse of the document frequency in which a word appears. TF-IDF is high if term t occurs many times in a small set of documents, and low if term t occurs several times in a document or very frequently in a large number of documents. Effective way to measure the similarity between document and query vector is a vector matching operation such as cosine similarity. This operation is used to measure the cosine of angle between vectors, and the smaller the resulting value, the more similar it is considered. The similarity result is used to rank the documents according to their estimated relevance to the query.

2.1.3 Language Model

The Language Model approach [1, 2] treats the process of document retrieval as probabilistic inference. That is, it builds a model from each document and then ranks the documents based on the probability that the document model generated a query. This document model generates queries using probabilistic estimates by Bayes theorem, n-grams [20], and so on. Although such a generative model for information retrieval is successful in modeling features such as text information, it is difficult to rank documents based on various features such as link and click information, which are obtained from search engine log data. In order to properly weight these various features, learning-to-rank [3] has been actively studied.

2.2 Learning-to-Rank

Learning-to-Rank refers to the application of supervised machine learning techniques to build a ranked model for information retrieval systems. It uses machine learning technologies to train the ranking model and has shown great value in many fields, such as document retrieval [3], recommender system [21] and question answering [22]. In the document retrieval, the learning-to-rank method learns an optimal way of combining various features extracted from query-document pairs through discriminative training. These features include features of the document itself that do not depend on the query, such as the number of in-out links and PageRank [23], features that depend on the query, such as the length and frequency of the query, and features that depend on both the query and the document, such as BM25 and TF-IDF. Given a query, the ranking model assigns a score to each document, and ranks the documents in descending order of the scores. The score represents the relevance of documents with respect to the query. The goal is to rank highly relevant documents higher than less relevant documents. Figure 2.1 shows a typical learning-to-rank architecture. The training data consists of several pairs of queries and their associated document sets, and each query-document pair has a label indicating its relevance. The ranking model is trained by each machine learning algorithm using the training data, and ranks and sorts the test data for which the ranking is not known. The learning-to-rank approach can be classified into three categories according to the learning method: pointwise, pairwise, and listwise methods.

2.2.1 Pointwise

In the pointwise approach, given a query group, a score is computed for each document independent of the other documents. The loss for each document is defined as the difference between its labeled score and its predicted score. The ranking of the result list is obtained

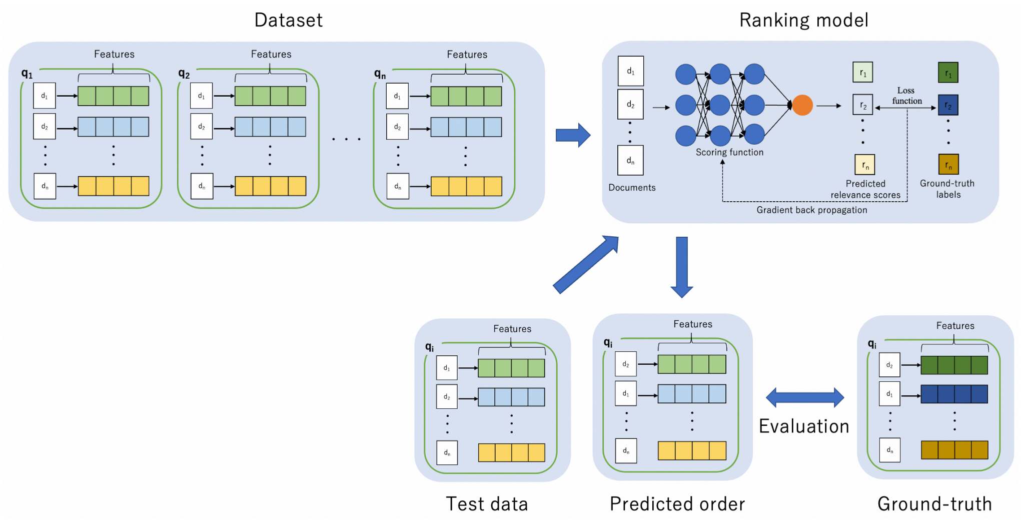


Figure 2.1: Architecture of learning-to-rank

by simply sorting the documents by their predicted scores. A pointwise approach trains a classifier or regressor that predicts the relevance of a document given a query, such as standard ML models. The disadvantage of the pointwise model is that it can only predict the relevance of a single document, making it difficult to rank it in relation to other documents obtained from the same query. Typical pointwise models are PRank [24] and McRank [25].

2.2.2 Pairwise

In the pairwise model, each document in a query is paired with other documents in the same query and fed into a scoring function to generate priorities among pairs of documents. After all the pairs are compared, the overall order of the documents is obtained. The loss function is set to minimize the number of pairs with wrong priorities between pairs of documents. This model can rank documents by considering the relative order of document pairs. Typical pointwise models are RankNet [26] and RankSVM [27].

2.2.3 Listwise

A listwise approach takes a list of all documents as input and tries to re-rank them an optimal order. Many listwise approaches learn to directly optimize metrics that evaluate rankings, such as Normalized Discounted Cumulative Gain (NDCG). Also, listwise approaches have been found to perform better than other approaches [28]. Typical pointwise models are ListNet [28] and LambdaRank [29].

These discriminative models can combine various features to output the best ranking, but it is difficult to obtain useful features from massive unlabeled documents. Therefore, there is an increasing movement to apply adversarial training, which trains generative models and discriminative models at the same time, to information retrieval.

2.3 Adversarial Training

Adversarial training is a learning method proposed by Generative Adversarial Nets (GAN) [6] that two models (Generator, Discriminator) improve themselves by beating each other at every round of this competition.

2.3.1 Generative Adversarial Nets

Figure 2.2 shows a typical architecture of GAN. The generator makes its own generation probability distribution close to the train data distribution and maximizes the probability that the discriminator makes a mistake. On the other hand, discriminator attempts to Identify whether it is a training sample or one sampled by the generator. The goal of GAN is to learn a distribution for the generator that approximates the distribution of real data. The optimization of GAN is done by the following minmax game using the joint loss function of discriminator and generator $V(G, D)$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

where z denotes noise, p_{data} is a ground-truth distribution and $p_z(z)$ is input noise variables. This machine learning framework is mainly used for image generation and data augmentation [30]. DCGAN is one of the most popular studies in image generation. It uses a convolutional layer to increase the expressive power of G, D, where a multilayer perceptron was used. The result of the generation was extremely innovative, and it has become possible to generate a fine image that is different from the conventional model. It is also possible to add and subtract images by computing the input vector. Recently, by making various learning techniques, it has become possible to generate a detailed image that can even be mistaken as a real image from noise [31].

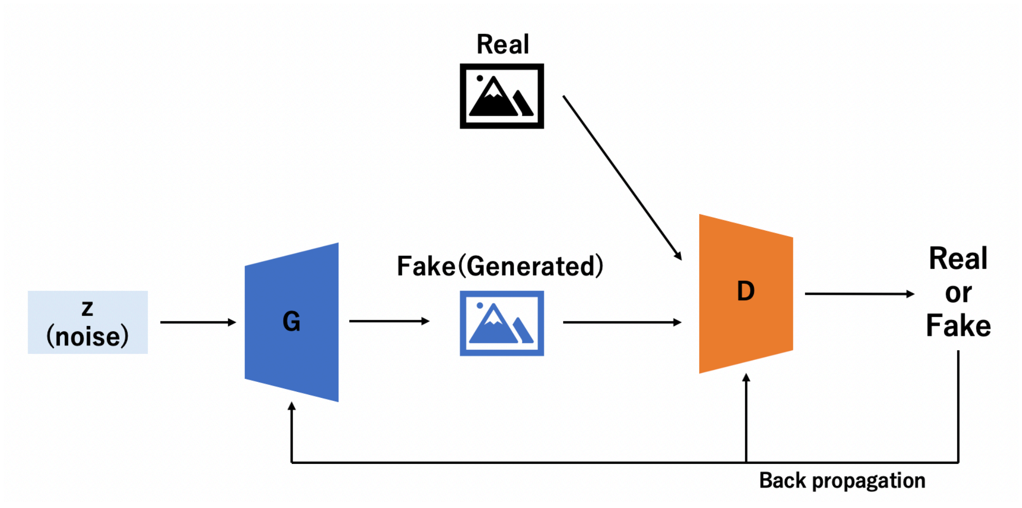


Figure 2.2: Architecture of Generative Adversarial Nets

2.3.2 f-GAN

f-GAN is one of GANs, which proposed generative neural samplers using variational divergence minimization [14]. In conventional GANs, GAN-divergence is used for optimization of the objective function, but in f-GAN, f-divergence, which is a generalization of the GAN-divergence, is used for learning. For a set χ of random variables x, f-divergence is defined as follows.

$$D_f(P||Q) = \int_{\chi} q(x) f\left(\frac{p(x)}{q(x)}\right) dx \tag{2.2}$$

where P and Q are distributions, and $p(x)$ and $q(x)$ denote density function. Here, from the divergence identity, f is a function satisfying $f(1) = 0$. For example, if $f(x) = xlog(x)$ gives KL-divergence, and $f(x) = xlog(x) - (x+1)log(x+1)$ gives the GAN-divergence. The variational divergence estimation framework is trained based on the following objective function.

$$F(\theta, \omega) = \mathbb{E}_{x \sim P}\left[g_f\left(V_\omega(x)\right)\right] + \mathbb{E}_{x \sim Q_\theta}\left[-f^*\left(g_f\left(V_\omega(x)\right)\right)\right] \tag{2.3}$$

where θ and ω represent the parameters of the generator and discriminator, respectively, V_ω is the output of the discriminator, g_f is the activation function of the output layer, and f^* is the convex conjugate function. It was shown that the model trained for KL-divergence achieved a higher holdout likelihood compared to the GAN model.

2.4 Divergence

The divergence is used as a measure of the difference between different probability distributions, and it weakens the axiom of distance, requiring only that non-negativity and Identity of indiscernible be satisfied. In this thesis, the following five divergences were used to train the proposed method.

2.4.1 Kullback-Leibler

Initially, the entropy with respect to a random variable x is defined as follows using its probability distribution p(x).

$$I(p(x)) = -\int p(x)\log(p(x))dx \tag{2.4}$$

When approximating a complex probability distribution $p(x)$ with a simple probability distribution $q(x)$, replacing the $-log(p(x))$ part, which indicates the amount of information, with $-log(q(x))$, we obtain

$$I(q(x)) = -\int p(x)\log(q(x))dx \tag{2.5}$$

This is called cross-entropy, and it indicates the difficulty of predicting the distribution $q(x)$ of observed data generated under the assumption of a true probability distribution $p(x)$. To approximate the two probability distributions, taking the difference between the cross-entropy and the original entropy, we obtain

$$\begin{aligned} \left(-\int p(x)\log(q(x))dx\right) - \left(-\int p(x)\log(p(x))dx\right) \\ = \int p(x)(\log(p(x)) - \log(q(x)))dx \\ D_{KL}(P||Q) = \int p(x)\log\left(\frac{p(x)}{q(x)}\right)dx \end{aligned} \tag{2.6}$$

This is called relative entropy, which is the difference between the two probability distributions when they are replaced by a probability distribution, i.e., how close the two probability distributions are. Relative entropy is also called *Kullback-Leiber divergence (KL-divergence)* KL-divergence calculates something like the distance between two probability distributions.

2.4.2 Pearson χ^2

Pearson χ^2 -divergence [32] is the squared loss of KL-divergence and is expressed as

$$D_{PC}(P||Q) = \int \frac{(q(x) - p(x))^2}{p(x)} \mathrm{d}x \tag{2.7}$$

Pearson χ^2 -divergence is a measure used in goodness-of-fit tests to test whether the observed frequency distribution is the same as the theoretical distribution, and has similar properties to KL-divergence. Since the squared function in Pearson χ^2 -divergence is compatible with the least-squares method, it can be computed more efficiently than KL-divergence and is also known to be robust against outliers [33].

2.4.3 Jensen-Shannon

Jensen-Shannon-divergence [34] is a symmetry and smoothing of KL-divergence in order to overcome the difficulty that KL-divergence does not satisfy the symmetry that is the axiom of distance, and is expressed by the following formula.

$$\begin{aligned} D_{JS}(P||Q) &= \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M) \\ &= \frac{1}{2}\int (p(x)\log\frac{2p(x)}{p(x)+q(x)} + q(x)\log\frac{2q(x)}{p(x)+q(x)})\mathrm{d}x \end{aligned} \tag{2.8}$$

where $M = \frac{1}{2}(P+Q)$.

2.4.4 GAN

GAN-divergence [6] is the divergence used when learning GAN, and is represented by the following equation.

$$\begin{aligned} D_{GAN} &= -\log 4 + 2 \cdot D_{JS}(P||Q) \\ &= \int (p(x)\log\frac{2p(x)}{p(x)+q(x)} + q(x)\log\frac{2q(x)}{p(x)+q(x)})\mathrm{d}x - \log 4 \end{aligned} \tag{2.9}$$

In GAN training, when the probability distribution of the generator and the dataset match (at Nash equilibrium), the objective function of the GAN takes the value of $-log4$, so $-log4$ is added to JS-divergence.

2.4.5 Squared Hellinger

Unlike KL-divergence and Pearson χ^2 -divergence, Squared Hellinger-distance [35] can be treated as a distance because it satisfies the distance axioms of symmetry and trigonometric inequality, and thus can be stably learned. Squared Hellinger-divergence is expressed by the following formula.

$$D_{SH}(P||Q) = \int (\sqrt{p(x)} - \sqrt{q(x)})^2 \, \mathrm{d}x \tag{2.10}$$

2.5 Adversarial Training for Learning-to-Rank

In recent years, research on applying adversarial training to learning-to-rank has been actively conducted [36, 37, 38]. Among them, IRGAN and AdvIR are the representative examples of the application of adversarial training in web search.

2.5.1 IRGAN

In IRGAN [11], the minmax game theory used in GAN was applied to optimize the generative and discriminative models for information retrieval. Generative model learns to make the distribution of document sampling close to the true distribution of the dataset (so that it can sample documents with relevance), while the discriminative model identifies whether the sampled documents are from the dataset or sampled by the generative model. In IRGAN, the generative model performs probabilistic sampling from discrete data, while GAN deals with images, which are continuous data. In other words, IRGAN does not generate documents, but samples them. This model is trained based on the following objective function.

$$J^{G^*,D^*} = \underset{\theta}{min}\, \underset{\varphi}{max} \textstyle\sum_{n=1}^{N} (\mathbb{E}_{d\sim p_{true}(d|q_n,r)}[logD(d|q_n)] + \mathbb{E}_{d\sim p_\theta(d|q_n,r)}[log(1-D(d|q_n))]) \tag{2.11}$$

where θ and ϕ represent the parameters of the generator and discriminator, $p_\theta(d|q_n, r)$ is the probability distribution of sampling relevant documents from the set of documents for the query q_n, and $p_{true}(d|q_n, r)$ is the true relevance distribution of the set of documents.

2.5.2 AdvIR

Adversarial training for Information Retrieval (AdvIR) [12] also applies adversarial training to the training of the traditional ad-hoc information retrieval model. The model generates documents that are difficult to judge by adding adversarial perturbation on top of the existing documents. Incorporating adversarial sampling with adversarial training, this model can generate even more difficult negative examples based on the adversarially sampled negative examples. It uses Kullback-Leibler divergence in the objective function to reduce the difference in distribution caused by the adversarial perturbation so that the model's output does not change dramatically due to the adversarial perturbation. This model is trained based on the following objective function.

$$\begin{aligned}\sum_q \Big(&\mathbb{E}_{d\sim p_{\text{data}}\,(d|q,y=1)}\left[\mathrm{J}(q,d,y=1;\theta) + \mathrm{J}_{\mathrm{KL}}(q,d;\theta)\right] + \\ &\mathbb{E}_{d\sim p_\theta(d|q,y=1)}\left[\mathrm{J}(q,d,y=0;\theta) + \mathrm{J}_{\mathrm{KL}}(q,d;\theta)\right]\Big)\end{aligned} \tag{2.12}$$

where θ is a set of model parameter, and $y \in \{0, 1\}$ denotes relevance label. p_{data} is a distribution of positive example being selected from label data and p_θ is a distribution that a negative example is selected from unlabeled data. $J(q, d, y = 0, 1; \theta)$ and $J_{KL}(q, d; \theta)$ are expressed as follows;

$$\begin{gathered}
\mathrm{J}_{\mathrm{KL}}(q, d; \theta) = \mathrm{KL}\left[p(\cdot \mid q, d; \theta) \| p\left(\cdot \mid q + \eta_q, d + \eta_d; \theta\right)\right] \\
\mathrm{J}(q, d, y; \theta) = -\log p(y \mid q, d; \theta), \text{ where} \\
p(y = 1 \mid q, d; \theta) = \sigma\left(f_\theta(q, d)\right), p(y = 0 \mid q, d; \theta) = 1 - \sigma\left(f_\theta(q, d)\right)
\end{gathered}$$

where p is a conditional relevance distribution. η_q and η_q are adversarial perturbations for q and d, and σ is a sigmoid function, f_θ is a scoring function.

These adversarial learning-to-rank approaches use GAN-divergence to train the model, but it is not known whether it is optimal for document retrieval as well. In the field of image retrieval, PC-divergence and KL-divergence achieved higher image accuracy and high holdout likelihood than GAN-divergence [14, 13]. In addition, the difficulty of stable learning is a problem in IRGAN [15, 12], especially in the pairwise model, it has been confirmed that the performance of the generator decreases with learning. From these questions, we proposed a variational divergence estimation framework using f-divergence minimization approach on adversarial learning-to-rank.

Chapter 3

Proposed Method

In this chapter, we describe how to perform adversarial learning-to-rank based on variational divergence minimization.

3.1 The Variational Divergence Estimation Framework

Let $\mathcal{Q}$ and $\mathcal{D}$ be the query space and the document space, respectively. We use $\Phi : \mathcal{Q} \times \mathcal{D} \to \mathcal{Z} := \mathbb{R}^d$ to denote the mapping function Φ that generates a feature vector for a document under a specific query context, where $\mathcal{Z}$ represents the d-dimensional feature space and $\mathbb{R}$ is a set of whole real numbers. We use $\mathcal{T} := \mathbb{R}$ to denote the space of the ground-truth labels each document receives. Thus, for each query, we have a list of document feature vectors $\mathbf{x} = (x_1, ..., x_m) \in \mathcal{X} := \mathcal{Z}^m$ and a corresponding list $\mathbf{y}^* = (y_1^*, ..., y_m^*) \in \mathcal{Y} := \mathcal{T}^m$ of ground-truth labels. The subscript i like x_i or y_i^* denotes the ith-position in the list. In practice, we get independently and identically distributed (i.i.d) samples $\mathcal{S} = \{(\mathbf{x}_j, \mathbf{y}_j^*)\}_{j=1}^n$ from an unknown joint distribution $P(\cdot, \cdot)$ over $\mathcal{X} \times \mathcal{Y}$. We use $h : \mathbf{x} \to \mathbb{R}^m$ to denote the real-valued scoring function, which assigns each document a score. The scores of the documents associated with the same query, i.e., $\mathbf{y} = h(\mathbf{x}) = (h(x_1), h(x_2), ..., h(x_m))$, are used to sort the documents. In order to learn an optimal scoring function, many prior studies are conducted based on the principle of *empirical risk minimization* [3].

Inspired by [11], we cast the process of learning-to-rank as a game between two opponents: a *generator* and a *discriminator*. The generator aims to generate (or select) documents that look like the ground-truth relevant documents, which may fool the discriminator. On the other hand, the discriminator aims to make a clear distinction between the ground-truth relevant documents and the ones generated by its opponent generator. Figure 3.1 shows the framework of the proposed method, adversarial learning-to-rank based on variational divergence minimization. Based on the variational divergence minimization framework introduced in [14], we formulate the adversarial learning-to-rank as follows:

$$J^{G^*,D^*} = \min_{\theta} \max_{\phi} \sum_{n=1}^{N} (\mathbb{E}_{\pi \backsim P_{true}(\pi|q_n)}[g_f(D_\phi(\pi|q_n))] - \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))]) \quad (3.1)$$

where π represents a document, a single document d_i in the case of pointwise, or a pair of documents $\langle d_i, d_j \rangle$ in the case of pairwise. The generator G is denoted as $P_\theta(\pi|q_n)$ that aims to minimize the objective. On the one hand, the generator fits the true distribution over all documents per query $\pi \backsim P_{true}(\pi|q)$. On the other hand, it randomly samples documents in

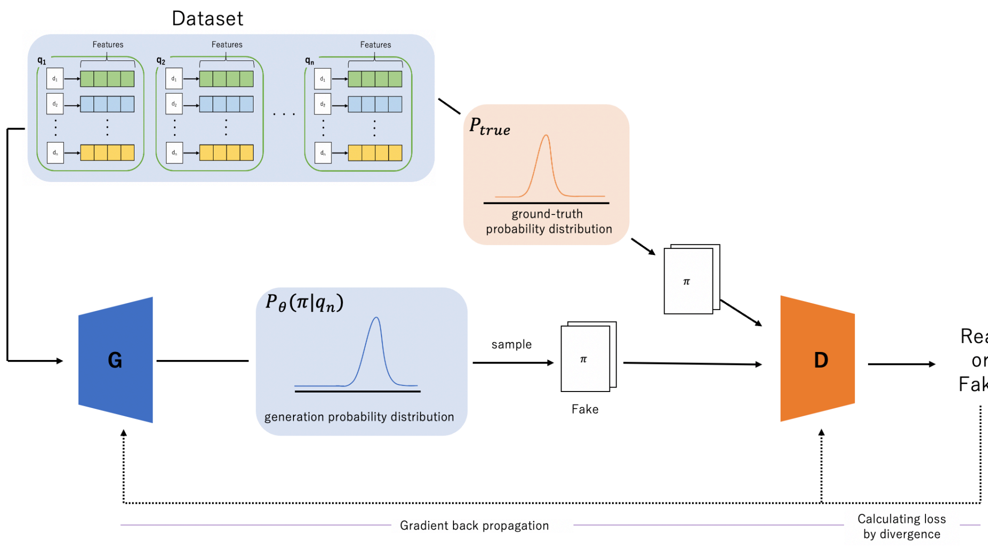

Dataset
Features
q_1
q_2
q_n
P_{true}
ground-truth probability distribution
π
G
$P_\theta(\pi|q_n)$
generation probability distribution
sample
Fake
D
Real or Fake
Gradient back propagation
Calculating loss by divergence

Figure 3.1: Architecture of the variational divergence estimation framework

order to fool the discriminator. The discriminator is denoted as $D_\phi(\pi|q_n)$, which estimates the probability of a document being either the ground-truth relevant document or not. f denotes the f-divergence, and f^* is the Fenchel conjugate of f. $g_f : \mathbb{R} \to dom_{f^*}$ is an output activation function which is specific to the adopted f-divergence. Table 3.1 shows a number of common instantiations of f-divergence, the corresponding conjugates and activation functions. In IRGAN training, after pre-training the generator and discriminator

Table 3.1: Parameter of each divergence

f-divergence f	Conjugate f^*	domain dom_{f^*}	output activation g_f
Kullback-Leibler	$exp(t-1)$	$\mathbb{R}$	v
Pearson χ^2	$\frac{1}{4}t^2 + t$	$\mathbb{R}$	v
Jensen-Shannon	$-log(2 - exp(t))$	$t < log(2)$	$log(2) - log(1 + exp(-v))$
Squared Hellinger	$\frac{t}{1-t}$	$t < 1$	$1 - exp(-v)$
GAN	$-log(1 - exp(t))$	$\mathbb{R}\_$	$-log(1 + exp(-v))$

with training data, training of the generator several epochs and training of the discriminator several epochs are repeated in order. This training procedure is difficult to propagate the gradient between the generator and the discriminator, so it can be unstable and prone to local optimum. In our study, we adopted the single-step gradient method [14], which calculates the gradient of θ and ϕ by a single backpropagation. To optimize discriminative and generative model, calculating the gradient of equation 3.1 with respect to ϕ, θ, then update. Finally, the optimal generator G^* and discriminator D^* are obtained by minimizing the variational divergence. We elaborate on the optimization of each opponent as below.

Algorithm 1 Single-step gradient method

```
Initialize generator and discriminator with random weights;
for number of epochs do
    Sample $X_{true} = (x_1, ..., x_N)$ from $P_{true}$.
    Sample $X_\theta = (x'_1, ..., x'_N)$ from $P_\theta$.
    D-step: Update $\phi$.
    $\phi^{t+1} = \phi^t + \nabla_\phi F(\theta^t, \phi^t)$.
    G-step: Update $\theta$.
    $\theta^{t+1} = \theta^t - \nabla_\theta F(\theta^t, \phi^t)$.
end for
```

3.2 Generator Optimization

3.2.1 The Pointwise Case

The generator G samples a document d based on its own generation probability distribution from a set of documents $\{d_1, ..., d_n\}$ that are candidates for query q. In our model, the goal of the generator G is to bring its own probability distribution closer to the ground truth probability distribution P_{true} by minimizing f-divergence. Thereby, when q is given to the generator, highly relevant d can be selected from the candidate pool. The ground-truth document given to the discriminator is randomly selected from the positive documents. On the other hand, fake documents are sampled by the generator. The probability that a particular document d_k is selected is then given by a softmax function.

$$P_\theta(\pi_k|q_n) = \frac{\exp(g_\theta(q, \pi_k))}{\sum_\pi \exp(g_\theta(q, \pi))} \tag{3.2}$$

where, g_θ shows a real-valued function reflecting the probability that document is selected from query. After fixing the discriminator, the generator is learned by minimizing equation (3.1) as follows:

$$\theta^* = \min_\theta \sum_{n=1}^{N} (\mathbb{E}_{\pi \backsim P_{true}(\pi|q_n)}[g_f(D_\phi(\pi|q_n))] - \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))]) \tag{3.3}$$

According to [6], maximizing $\mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[logD_\phi(x)]$ is used to speed up learning rather than minimizing $\mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[log(1 - D_\phi(x))]$, so following it, the generator was updated to maximize the θ in Algorithm 1 as follows:

$$\theta^{t+1} = \theta^t + \nabla_\theta \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))] \tag{3.4}$$

Since document sampling is discrete, unlike image generation, it cannot be optimized by gradient descent, so it is optimized by REINFORCE algorithm [39] according to [11]. The gradient is calculated as follows.

$$
\begin{aligned}
&\nabla_\theta \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))] \\
&= \sum_{i=1}^{M} \nabla_\theta P_\theta\left(\pi \mid q_n\right) f^*(g_f(D_\phi(\pi|q_n))) \\
&= \sum_{i=1}^{M} P_\theta\left(\pi \mid q_n\right) \nabla_\theta \log P_\theta\left(\pi \mid q_n\right) f^*(g_f(D_\phi(\pi|q_n))) \\
&= \mathbb{E}_{\pi \sim P_\theta(\pi|q_n)}\left[\nabla_\theta \log P_\theta\left(\pi \mid q_n\right) f^*(g_f(D_\phi(\pi|q_n)))\right] \\
&\simeq \frac{1}{K}\sum_{k=1}^{K} \nabla_\theta \log P_\theta\left(\pi_k \mid q_n\right) f^*(g_f(D_\phi(\pi_k|q_n))) \qquad (3.5)
\end{aligned}
$$

where m represents the total number of documents for a query, and π_k represents the k-th document sampled by the current generator $P_\theta(\pi|q_n)$. Eq 3.5 approximates sampling by a reinforcement learning algorithm.

3.2.2 The Pairwise Case

Here we show that our proposed framework also works in pairwise case. For each query q_n, we have a set of labelled document pairs $D_n = \{\langle d_i, d_j\rangle | d_i \geqq d_j\}$. The generator G tries to generate a pair of documents from D_n. In pairwise, pairs are made between sorted sets of documents of the same query, and each document pair is weighted by positions based on the number of positive and explicit documents in the query. Initially, the difference in the relevance of document pairs is weighted by the number of explicit documents as follows,

$$P_{true}\left(\pi|q_n\right) = \frac{p_i - p_j}{\log_2(i_e + 2)} \qquad (3.6)$$

p_i, p_j express the relevance of the document i, j and i_e denotes the i-th position in the explicit documents. Then weight them in the same way, according to the number of positive documents as follows,

$$P_{true}\left(\pi|q_n\right) = \frac{P_{true}\left(\pi|q_n\right)}{\log_2(i_p + 2)} \qquad (3.7)$$

i_p denotes the i-th position in the positive documents. Based on the probability distribution, ground-truth pairs are sampled. On the other hand, we sampled fake document pairs using the Bradley-Terry model [40]. It is a probabilistic model that can predict the result of pair comparison. Estimating the probability that $i > j$ given a pair of individuals i, j selected from a certain population. In our model, fake document pair sampling is performed based on the following probabilities.

$$P_\theta(i > j) = \frac{1}{1 + e^{-(p_i - p_j)}} \qquad (3.8)$$

As with pointwise case, after fixing the discriminator, the generator is learned as follows:

$$
\begin{aligned}
\theta^* &= \min_\theta \sum_{n=1}^{N} (\mathbb{E}_{\pi \backsim P_{true}(\pi|q_n)}[g_f(D_\phi(\pi|q_n))] - \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))]) \\
&= \max_\theta \sum_{n=1}^{N} (\mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))]) \qquad (3.9)
\end{aligned}
$$

where $\pi = \langle d_i, d_j\rangle$ denotes real document pairs for q_n.

3.3 Discriminator Optimization

3.3.1 The Pointwise Case

The discriminator $D_\phi(\pi|q_n)$ maximizes the likelihood of correctly discriminating d that obtained from the generator and ground-truth. After training the generator, a set of documents sampled by the generator and ground truth are given. Using these documents, the discriminator $D_\phi(\pi|q_n)$ is trained as a binary classifier that classifies a given document-query pair into either 0 (generated) or 1 (ground-truth). Therefore, the goal of the discriminator $D_\phi(\pi|q_n)$ is to distinguish correctly whether the document was sampled by the generator or ground truth. In the discriminative model, the sigmoid function is used to estimate the probability that a document d is related to a query q.

$$D_\phi(\pi \mid q_n) = \sigma\left(f_\phi(\pi, q_n)\right) = \frac{\exp\left(f_\phi(\pi, q_n)\right)}{1 + \exp\left(f_\phi(\pi, q_n)\right)} \tag{3.10}$$

where f_θ shows a real-valued function reflecting the probability that a document is generated from a query. After sampling documents by ground-truth and the generator, the probability of each ground-truth document and fake document are estimated by the discriminator, and the discriminator's loss is calculated by the activation function and Fenchel conjugate for each divergence. The activation function is applied to ground-truth documents, and the activation function and Fenchel conjugate are applied to fake documents, i.e. the discriminator is learned as follows.

$$\phi^* = \max_\phi \sum_{n=1}^{N} (\mathbb{E}_{\pi \backsim P_{true}(\pi|q_n)}[g_f(D_\phi(\pi|q_n))] - \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))]) \tag{3.11}$$

3.3.2 The Pairwise Case

After the generator G has sampled the document, the discriminator $D_\phi(\pi|q_n)$ tries to distinguish such a generated document pair from a real document pair. As with the pointwise case, after sampling documents by ground-truth and generator G, the discriminator is learned as follows:

$$\phi^* = \max_\phi \sum_{n=1}^{N} (\mathbb{E}_{\pi \backsim P_{true}(\pi|q_n)}[g_f(D_\phi(\pi|q_n))] - \mathbb{E}_{\pi \backsim P_\theta(\pi|q_n)}[f^*(g_f(D_\phi(\pi|q_n)))]) \tag{3.12}$$

where $\pi = \langle d_i, d_j \rangle$ denotes real document pairs for q_n and $D_\phi(\langle d_i, d_j \rangle | q) = D_\phi(d_i, q) - D_\phi(d_j, q)$

Chapter 4

Experiment

In this chapter, we describe the experimental setup. We first introduce the data collection and the way of evaluation. Finally, we explain the structure of the baseline and proposed model.

4.1 Dataset

In our experiments, we used four publicly available learning-to-rank datasets, MQ2008 [41], MQ2008-semi [41], MSLR-WEB30K [42], and Yahoo Learning to Rank Challenge corpus [43].

4.1.1 MQ2008

MQ2008 is a query set from Million Query track of TREC 2008 provided by Microsoft, which are crawled from Web sites in the .GOV domain. This dataset consists of feature vectors extracted from query-url pairs along with relevance judgment labels. The ground truth is a multiple-level relevance judgment, which takes 3-level values from 0 to 2. This dataset contains more than 784 queries, and the query-url pair is represented by a 46-dimensional feature vector.

4.1.2 MQ2008-semi

MQ2008-semi is designed for semi-supervised learning. This dataset consists of a small number of pairs of labeled data and a large number of pairs of unlabeled data. There are 15,000 labeled query-document pairs and 531,050 unlabeled query-document pairs associated with 784 queries. The relevance of the documents is judged 4-level values from -1 to 2. -1 means “ unknown ”, i.e., this query-document pairs are unlabeled examples. Each query-url pair is also represented by a 46-dimensional feature vector.

4.1.3 MSLR-WEB30K

MSLR-WEB30K is a large scale dataset for research on learning to rank provided by Microsoft. This dataset contains queries and documents sampled from real search engines (Microsoft Bing), and the relevance of the documents is judged 5-level values from 0 (irrelevant) to 4 (perfectly relevant). This dataset contains more than 30,000 queries, and the query-url pair is represented by a 136-dimensional feature vector.

4.1.4 Yahoo Learning to Rank Challenge corpus

Yahoo Learning to Rank Challenge corpus (Yahoo!LETOR) is a learning-to-rank collection published by Yahoo!. Yahoo!LETOR consists of two collections: Set 1 and Set2. In this work, we use only Set1. Set 1 was built based on the US Yahoo! search engines, and the relevance of the documents is also judged at 5-level values from 0 (irrelevant) to 4 (perfectly relevant). This dataset contains more than 29,921 queries, and the query-url pair is represented by a 700-dimensional feature vector.

MSLR-WEB30K, MQ2008 and MQ2008-semi are partitioned into five folds, and we performed five-fold cross validation. In our experiment, we use training, validation and test data. The training data is used to learn the ranking model, validation data is used to find the best hyperparameters, and the testing data for evaluation. Relevance varies depending on the dataset, but when evaluating, normalization is performed by nDCG.

Table 4.1: Properties of the four benchmark datasets

	MQ2008	MQ2008-Semi	MSLR-WEB30K	Yahoo(Set1)
Queries	784	784	31,531	29,921
Documents	15,211	546,260	3,771,125	709,877
Features	46	46	136	700
Relevance labels	{0, 1, 2}	{-1, 0, 1, 2}	{0, 1, 2, 3, 4}	{0, 1, 2, 3, 4}
Avg relevant documents/query	3.7	3.7	58.0	17.5
Avg documents/query	19.4	696.8	119.6	23.7

4.2 Evaluation Metrics

We use nDCG to measure the performance. It is normalized by dividing the Discounted Cumulative Gain (DCG) [44] obtained from the predicted ranking of the ranking model by the DCG obtained from the ideal ranking. It takes into account the ranking position of the item, where a higher position is assigned with a higher score. It is calculated as follow: Given a query, nDCG is defined as

$$\mathrm{nDCG@k} = \frac{\mathrm{DCG@k}}{\mathrm{IDCG@k}} \tag{4.1}$$

where k is the cutoff value, which indicates the number of elements used for evaluation. DCG is defined as

$$\mathrm{DCG@k} = \sum_{i=1}^{k} \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+2)} \tag{4.2}$$

where, rel_i is the degree of relevance of the document at position i in the ranking list, which is suggested by ranking model. IDCG is defined as

$$\mathrm{IDCG@k} = \sum_{i=1}^{k} \frac{2^{\mathrm{ideal}_i} - 1}{\log_2(i+2)} \tag{4.3}$$

where, $ideal_i$ is the degree of relevance of the document at position i if we rank the matched documents by their ground truth relevance. We report the results with different cutoff values 1, 3, 5 and 10 to show the performance of method at different positions and the ranking performance based on the averaged nDCG scores across five folds with 100 epochs.

4.3 Baseline Method and Model Configuration

We compare the proposed model (IRf-GAN) with IRGAN, pointwise and pairwise respectively. For the fair comparison, the structure of each model and other hyperparameter settings such as learning rate were set to the same value. Each model consists of simple 5-layer feed-forward neural network (a dropout rate of 0.01) and the dimension of a hidden layer is set as 100. We used the L2 regularization with a decaying rate of 0.001 and the Adam [45] optimizer with a learning rate of 0.001. In order to find a more accurate model, we experimented with three activation functions for each layer: Sigmoid, Relu and CELU. In our proposed method, the divergence was selected from the divergence functions, Kullback-Leibler, Pearson χ^2, Jensen-Shannon, GAN, and Squared Hellinger.

Chapter 5

Results and Discussion

In this chapter, we show the results of the involved methods on each dataset, and then discuss the results by comparing different methods.

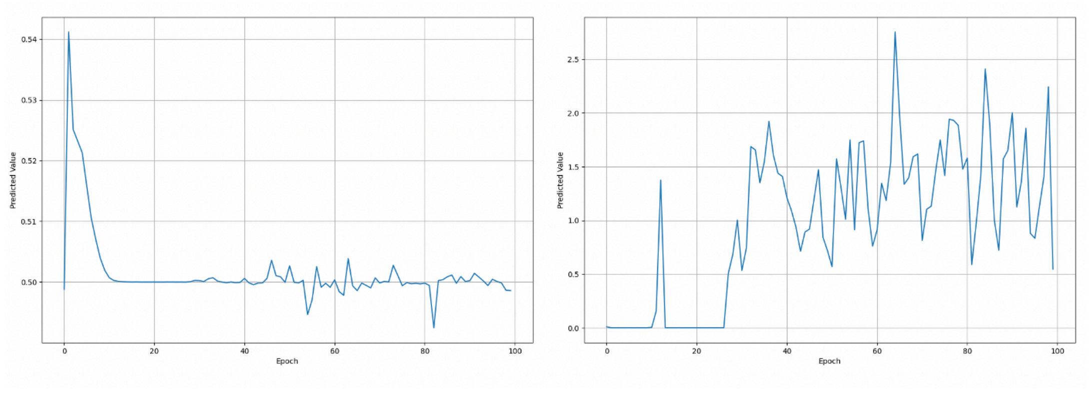
5.1 Results and Analysis

In this section, we first show the results of pointwise methods, and then the results of pairwise methods.

5.1.1 MQ2008 dataset

Pointwise

Table 5.1 show the values of nDCG@k(k=1,3,5,10) for pointwise method of IRGAN and IRf-GAN respectively. It is observed that the IRf-GAN outperforms the IRGAN at all divergences. In particular, the performance is high when using the GAN-divergence, with the highest values for all cases from nDCG@1 to nDCG@10. Comparing the IRGAN score with the highest score among the IRf-GANs, there was an improvement of about 10.5%, indicating a significant difference. When the activation function was set to Relu or CELU in IRGAN, the prediction of the generator started to show nan in the middle of training, and training could not be performed. It is probably because the output of the generator became extremely small as the learning progressed and the loss of the generator diverged. As a result, the generator will not be able to sample fake documents well, and the discriminator will not be able to learn well. Therefore, in IRGAN learning, it is necessary to apply the sigmoid function to each layer to limit the range of output values for stable learning. IRf-GAN did not fail to learn, but when the activation function was set to sigmoid function, the score was lower than the other activation functions. The same problem occurred on other datasets (MQ2008-semi, MSLR-WEB30K). One possible reason is that by repeatedly applying the sigmoid function, the output of the generator and discriminator only be the same value, and learning does not proceed. Figure 5.1 shows a plot of the predicted value of the generator when learning with GAN-divergence. The Figure 5.1(a) shows the change in the predicted value when sigmoid is used as the activation function, and the Figure 5.1(b) shows the change in the predicted value when Relu is used. We can see that the predicted value converges to 0.5 early in the learning process. It is likely that the learning failure is caused by the convergence of generator's prediction.



(a)Activation function : Sigmoid (b)Activation function : Relu

Figure 5.1: Changes in the value of prediction of generator, where the x-axis represents number of epochs, and the y-axis represents the change of predicted value of generator.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.4056	0.4271	0.4647	0.5591
		R	0.3495	0.3808	0.4122	0.4774
		CE	0.0848	0.0904	0.0989	0.1175
IRf-GAN	KL	S	0.2965	0.3367	0.3712	0.4478
		R	0.4473	0.4623	0.5149	0.5990
		CE	0.3989	0.4580	0.5062	0.5821
	JS	S	0.2110	0.2508	0.2887	0.3821
		R	0.4249	0.4531	0.4882	0.5744
		CE	0.4259	0.4544	0.4946	0.5784
	PC	S	0.2853	0.3050	0.3370	0.4183
		R	0.4272	0.4518	0.4911	0.5787
		CE	0.4261	0.4652	0.5045	0.5860
	GAN	S	0.2746	0.3069	0.3343	0.4308
		R	**0.4524**	**0.4771**	0.5121	**0.5988**
		CE	0.4290	0.4687	**0.5180**	0.5916
	SH	S	0.3093	0.3269	0.3772	0.4661
		R	0.3902	0.4481	0.4887	0.5744
		CE	0.4222	0.4549	0.5026	0.5844
Impv-IRf-GAN			11.5%	11.7%	11.5%	7.1%

Table 5.1: The experimental results of pointwise method on the MQ2008 dataset. The best result is indicated in bold, and the one that failed to learn in the middle is indicated in underline. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

Pairwise

Table 5.2 show the performance for pairwise method of IRGAN and IRf-GAN respectively. We can observe that IRf-GAN achieved better performance than IRGAN. Except for KL-divergence and SH-divergence, the other divergences had stable and high performance, especially JS-divergence scored highest in nDCG@1 and GAN-divergence scored highest in nDCG@3,5,10. The improvement rate was lower than that of the pointwise method, the average improvement rate was 1.1%, which was not enough to call it an improvement.

Considering the results of the pointwise method, We can say that JS-divergence and GAN-divergence are the best way to approximate probability distributions in this dataset.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.4592	0.4805	0.5293	0.6079
		R	0.4078	0.4410	0.4849	0.5737
		CE	0.4439	0.4631	0.5028	0.5802
IRf-GAN	KL	S	0.4373	0.4674	0.5177	0.6010
		R	0.4009	0.4373	0.4752	0.5600
		CE	0.4374	0.4803	0.5273	0.6027
	JS	S	0.4529	0.4732	0.5211	0.6039
		R	**0.4707**	0.4857	0.5188	0.6029
		CE	0.4474	0.4799	0.5178	0.5989
	PC	S	0.4509	0.4774	0.5221	0.6071
		R	0.4422	0.4702	0.5201	0.5971
		CE	0.4548	0.4727	0.5212	0.6009
	GAN	S	0.4563	0.4854	0.5289	0.6073
		R	0.4610	**0.4858**	0.5228	0.6027
		CE	0.4590	0.4847	**0.5313**	**0.6103**
	SH	S	0.4433	0.4726	0.5183	0.6069
		R	0.4294	0.4568	0.4990	0.5873
		CE	0.4409	0.4632	0.5179	0.6003
Impv-IRf-GAN			2.5%	1.1%	0.4%	0.4%

Table 5.2: The experimental results of pairwise method on the MQ2008 dataset. The best result is indicated in bold. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

5.1.2 MQ2008-semi dataset

Pointwise

Table 5.3 show the performance of pointwise method of IRGAN and IRf-GAN on the MQ2008-semi dataset. We can observe that IRf-GAN with JS-divergence performed better for nDCG@1, 3, 5, 10. In IRf-GAN, there was not much difference in performance between divergences, but relatively high performance was achieved when approximated by JS-divergence. Although the improvement rate of nDCG@1 and nDCG@3 is low (1.6% and 4.4%), it increases as the number of documents used for ranking increases, and the average of nDCG@1, 3, 5, and 10 is 10.2%, which can be concluded as an improvement.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.4389	0.4664	0.4687	0.4923
		R	<u>0.0000</u>	<u>0.0000</u>	<u>0.0000</u>	<u>0.0000</u>
		CE	<u>0.0000</u>	<u>0.0000</u>	<u>0.0000</u>	<u>0.0000</u>
IRf-GAN	KL	S	0.2096	0.2756	0.2978	0.3617
		R	0.4452	0.4794	0.5135	0.5903
		CE	0.4294	0.4720	0.5144	0.5983
	JS	S	0.1923	0.2462	0.2684	0.3273
		R	**0.4457**	**0.4870**	**0.5237**	**0.6053**
		CE	0.4259	0.4710	0.5170	0.5905
	PC	S	0.2331	0.3097	0.3425	0.4045
		R	0.4235	0.4653	0.5101	0.5899
		CE	0.4115	0.4595	0.5038	0.5875
	GAN	S	0.1495	0.2367	0.2621	0.3139
		R	0.4372	0.4558	0.5036	0.5866
		CE	0.4283	0.4804	0.5197	0.5970
	SH	S	0.2585	0.3168	0.3385	0.3959
		R	0.4361	0.4769	0.5170	0.5993
		CE	0.4263	0.4650	0.5074	0.5879
Impv-IRf-GAN			1.6%	4.4%	11.7%	23.0%

Table 5.3: The experimental results of pointwise method on the MQ2008-semi dataset. The best result is indicated in bold, and the one that failed to learn in the middle is indicated in underline. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

Pairwise

Table 5.4 shows the performance of pairwise method of IRGAN and IRf-GAN on the MQ2008-semi dataset. For the pairwise method, IRf-GAN outperformed IRGAN when using JS-divergence and GAN-divergence. It was observed that GAN-divergence is the best choice for nDCG@1,3,5 and JS-divergence is the best choice for nDCG@10. The improvement rates for nDCG@1 and 3 were relatively high at 7.8% and 3.9%, while the improvement rates for nDCG@5 and 10 were low, average was about 4.1%. The possibility of error cannot be denied, so it would not be possible to say that there is a significant difference in performance between IRGAN and IRf-GAN. The results of the MQ2008-semi dataset are similar to those of MQ2008, and even in this dataset, the performance of both pointwise and pairwise methods is superior when JS-divergence and GAN-divergence.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.4375	0.4776	0.5202	0.6001
		R	0.4378	0.4880	0.5163	0.5976
		CE	0.4365	0.4816	0.5241	0.6062
IRf-GAN	KL	S	0.4359	0.4856	0.5207	0.6028
		R	0.4437	0.4797	0.5260	0.5982
		CE	0.4324	0.4917	0.5236	0.6022
	JS	S	0.4416	0.4860	0.5303	0.6063
		R	0.4478	0.4833	0.5259	0.6049
		CE	0.4686	0.4977	0.5417	**0.6136**
	PC	S	0.4424	0.4790	0.5204	0.6006
		R	0.4442	0.4915	0.5296	0.6068
		CE	0.4573	0.4810	0.5219	0.5947
	GAN	S	0.4472	0.4887	0.5313	0.6087
		R	0.4658	0.4947	0.5353	0.6093
		CE	**0.4721**	**0.5072**	**0.5421**	0.6131
	SH	S	0.4402	0.4885	0.5289	0.6042
		R	0.4480	0.4923	0.5328	0.6019
		CE	0.4685	0.4936	0.5385	0.6128
Impv-IRf-GAN			7.8%	3.9%	3.4%	1.2%

Table 5.4: The experimental results of pairwise method on the MQ2008-semi dataset. The best result is indicated in bold. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

5.1.3 MSLR-WEB30K dataset

Pointwise

Table 5.5 shows the performance of pointwise method of IRGAN and IRf-GAN on the MSLR-WEB30K dataset. From the table, we can observe that IRf-GAN outperforms IRGAN in terms of all nDCG@1,3,5,10. Among them, JS-divergence is the best for nDCG@1, and PC-divergence is the best for nDCG@3,5,10. However, the improvement rate of IRf-GAN was low, averaging less than 1.0%, indicating that there was not a significant difference in performance.

Pairwise

Table 5.6 shows the performance of pairwise method of IRGAN and IRf-GAN on the MSLR-WEB30K dataset. It was confirmed that IRf-GAN outperformed IRGAN at any divergence was used. In particular, when training with PC-divergence, the performance was better for all nDCG@1,3,5,10, no matter which activation function was used in the model. Unlike the pointwise method, the IRf-GAN performance improvement rate is high, averaging 6.4%, which can be said to be a significant difference. Considering the pointwise results, the optimal divergence differs from the MQ2008 dataset and the MQ2008-semi dataset, and PC-divergence is better for large dataset such as MSLR-WEB30K.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.3879	0.3809	0.3852	0.4022
		R	0.4121	0.3990	0.4023	0.4177
		CE	<u>0.0000</u>	<u>0.0000</u>	<u>0.0000</u>	<u>0.0000</u>
IRf-GAN	KL	S	0.1450	0.1595	0.1726	0.2016
		R	0.3026	0.3114	0.3238	0.3522
		CE	0.3338	0.3373	0.3469	0.3715
	JS	S	0.1390	0.1550	0.1690	0.1995
		R	0.4094	0.3946	0.3970	0.4124
		CE	**0.4170**	0.4002	0.4029	0.4177
	PC	S	0.1387	0.1552	0.1690	0.1985
		R	0.4169	**0.4013**	**0.4041**	**0.4191**
		CE	0.2182	0.2244	0.2344	0.2547
	GAN	S	0.1388	0.1546	0.1688	0.1989
		R	0.4114	0.3931	0.3954	0.4111
		CE	0.4148	0.3963	0.3980	0.4132
	SH	S	0.1402	0.1552	0.1695	0.1998
		R	0.4061	0.3921	0.3943	0.4094
		CE	0.4100	0.3934	0.3961	0.4124
Impv-IRf-GAN			1.2%	0.6%	0.5%	0.4%

Table 5.5: The experimental results of pointwise method on the MSLR-WEB30K dataset. The best result is indicated in bold, and the one that failed to learn in the middle is indicated in underline. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.3970	0.3887	0.3938	0.4113
		R	0.4210	0.4029	0.4052	0.4175
		CE	0.4319	0.4142	0.4179	0.4334
IRf-GAN	KL	S	0.4571	0.4353	0.4379	0.4519
		R	0.4574	0.4380	0.4410	0.4558
		CE	0.4573	0.4381	0.4408	0.4552
	JS	S	0.4328	0.4134	0.4172	0.4320
		R	0.4472	0.4261	0.4306	0.4463
		CE	0.4513	0.4297	0.4320	0.4458
	PC	S	0.4612	0.4388	0.4407	0.4547
		R	**0.4643**	**0.4417**	**0.4434**	**0.4571**
		CE	0.4628	0.4408	0.4426	0.4562
	GAN	S	0.4320	0.4146	0.4175	0.4326
		R	0.4490	0.4276	0.4312	0.4464
		CE	0.4484	0.4274	0.4306	0.4452
	SH	S	0.4364	0.4171	0.4203	0.4356
		R	0.4481	0.4281	0.4314	0.4456
		CE	0.4530	0.4302	0.4328	0.4474
Impv-IRf-GAN			7.5%	6.6%	6.1%	5.5%

Table 5.6: The experimental results of pairwise method on the MSLR-WEB30K dataset. The best result is indicated in bold. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

5.1.4 Yahoo dataset

Pointwise

Table 5.7 shows the performance of pointwise method of IRGAN and IRf-GAN on the Yahoo dataset. IRf-GAN achieved better performance than IRGAN for all nDCG@1,3,5,10, and showed the best performance when approximated by PC-divergence on this dataset as well. The improvement in performance of IRf-GAN was low, averaging about 2.0%, which was not a significant difference. Compared to the results of other datasets, the difference in nDCG scores by divergence was large, and the performance of IRf-GANs trained with divergences other than KL-divergence and PC-divergence was low. In this dataset, the percentage of relevant documents in the set of documents for each query is larger than in other datasets. Therefore, it is probable that the generator sampled the related document in the early stage of learning and successfully deceived the discriminator, so that the same document was sampled after that, and the learning did not proceed. So it indicates that it is important to choose the appropriate divergence.

Pairwise

Table 5.8 shows the performance of pairwise method of IRGAN and IRf-GAN on the Yahoo dataset. IRf-GAN's performance far exceeded IRGAN's performance. It stably achieved good performance no matter which divergence it was approximated with, similar to the pointwise method, and was especially better when approximated with PC-divergence. The performance improvement was also very high for nDCG@1, 3, 5, and 10, at 17.8%, 11.0%, 8.9%, and 6.1%, averaging about 11.0%, which was the highest improvement across all datasets. Similar to the results of the MSLR-WEB30K dataset, the Yahoo dataset also suggests that approximating the probability distribution by PC-divergence is optimal for both the pointwise and pairwise methods.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.3296	0.3699	0.4067	0.4920
		R	0.3445	0.3947	0.4349	0.5204
		CE	0.5219	0.5490	0.5771	0.6429
IRf-GAN	KL	S	0.3230	0.3636	0.4027	0.4891
		R	0.5164	0.5464	0.5734	0.6400
		CE	0.5033	0.5367	0.5675	0.6360
	JS	S	0.3230	0.3636	0.4027	0.4891
		R	0.4302	0.4808	0.5188	0.5920
		CE	0.3839	0.4326	0.4778	0.5628
	PC	S	0.3230	0.3636	0.4027	0.4891
		R	**0.5371**	**0.5640**	**0.5866**	**0.6468**
		CE	0.4828	0.5197	0.5507	0.6207
	GAN	S	0.3230	0.3636	0.4027	0.4891
		R	0.4150	0.4652	0.5051	0.5828
		CE	0.4207	0.4678	0.5083	0.5868
	SH	S	0.3230	0.3636	0.4027	0.4891
		R	0.3804	0.4300	0.4723	0.5574
		CE	0.4054	0.4556	0.4973	0.5770
Impv-IRf-GAN			2.9%	2.7%	1.6%	0.6%

Table 5.7: The experimental results of pointwise method on the Yahoo dataset. The best result is indicated in bold. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

	Divergence	Activation function	nDCG@1	nDCG@3	nDCG@5	nDCG@10
IRGAN		S	0.5605	0.5852	0.6098	0.6708
		R	0.5652	0.5850	0.6092	0.6693
		CE	0.5494	0.5739	0.5998	0.6625
IRf-GAN	KL	S	0.6565	0.6465	0.6610	0.7090
		R	0.6534	0.6436	0.6550	0.7042
		CE	0.6581	0.6454	0.6599	0.7068
	JS	S	0.6441	0.6297	0.6442	0.6951
		R	0.6563	0.6436	0.6569	0.7069
		CE	0.6369	0.6278	0.6438	0.6941
	PC	S	**0.6660**	**0.6495**	**0.6638**	**0.7116**
		R	0.6589	0.6474	0.6625	0.7113
		CE	0.6615	0.6450	0.6591	0.7094
	GAN	S	0.6362	0.6272	0.6439	0.6947
		R	0.6543	0.6408	0.6576	0.7066
		CE	0.6540	0.6416	0.6578	0.7058
	SH	S	0.6457	0.6326	0.6467	0.6972
		R	0.6607	0.6439	0.6576	0.7053
		CE	0.6368	0.6229	0.6357	0.6854
Impv-IRf-GAN			17.8%	11.0%	8.9%	6.1%

Table 5.8: The experimental results of pairwise method on the Yahoo dataset. The best result is indicated in bold. The last line shows the improvement rate calculated based on the IRGAN score and the highest score among the IRf-GANs.

5.2 Discussion

Comparing the performance of IRf-GAN and IRGAN

The results for all four major learning-to-rank datasets showed that IRf-GAN outperformed IRGAN in performance. In particular, the pairwise method of IRf-GAN achieved stable and high performance. The possible reasons are as follows: First, the difference of the learning method. In IRGAN, the generator is trained for several epochs and then the discriminator is trained for several epochs. This process is repeated many times, so it is difficult to back-propagate the gradients of the generator and discriminator, making it difficult to stabilize the learning. On the other hand, IRf-GAN uses the single-step gradient method for learning, which eliminates the learning instability of IRGAN and achieves higher performance. Second, the difference of sampling method is thought to have an effect. In the pairwise method in IRf-GAN, when ground-truth documents are sampled, the sampling probability distribution is weighted according to the number of positive documents and the number of explicit documents in the query. In addition, when sampling fake documents, we consider the pairwise relation based on the Bradley-Terry model. This makes it difficult for the discriminator to identify whether a given document pair is sampled by the generator or ground-truth. As a result, the competition between the two models went well and each model achieved good performance. Fig 5.2 shows the learning curve of IRGAN and IRf-GAN on the MQ2008 dataset. In IRGAN, it can be observed that the performance of the discriminator improves with learning, but the performance of the generator does not change much. On the other hand, in IRf-GAN, the performance of both the discriminator and the generator improves with learning, suggesting that IRf-GAN is more stable in learning.

When we observed the improvement rate of IRf-GAN in each dataset, we found that the pointwise method was more significant for MQ2008 and MQ2008-semi dataset, while the pairwise method was more significant for MSLR-WEB30K and Yahoo dataset. This suggests that the importance of pair sampling increases for datasets with large number of queries, documents, and features.

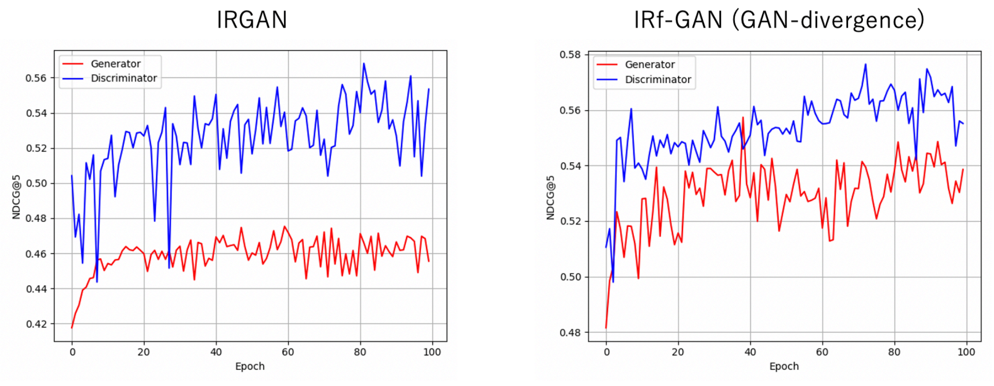


Figure 5.2: Learning curve of IRGAN and IRf-GAN on MQ2008 dataset, where the x-axis represents number of epochs, and the y-axis represents the nDCG@5 scores of the generator and discriminator, respectively.

Analyzing the effect of each divergence on each dataset

What is interesting throughout the experiment with the four major learning-to-rank datasets is that the optimal divergence was clearly divided between the results of MQ2008 and MQ2008-semi dataset and the results of MSLR-WEB30K and Yahoo dataset. It was shown that JS and GAN-divergence are optimal for MQ2008 and MQ2008-semi datasets, and PC-divergence is optimal for MSLR-WEB30K and Yahoo datasets. One possible reason for the difference in optimal divergence among the datasets is the number of features in the dataset. When comparing the results of the MSLR-WEB30K and Yahoo datasets, the Yahoo dataset has a higher rate of performance improvement due to PC-divergence. This means that as the number of features increases, PC-divergence becomes more effective. This may be due to the fact that PC-divergence can produce overdispersed approximations [13]. Since the distribution of documents is dispersed as the features increase, PC-divergence with such properties can be approximated better. In addition, in the experiment on Yahoo dataset, the pointwise method of IRf-GAN sometimes failed to learn well due to divergence, suggesting that it is important to choose the appropriate divergence.

5.3 Summary

In this section, we investigate the effectiveness of the proposed method (IRf-GAN), which is an adversarial learning-to-rank framework based on variational divergence minimization, on four major learning-to-rank datasets. Our main findings is that: 1) IRf-GAN achieved better performance than IRGAN on most of the datasets, and the fact that IRf-GAN, which approximates the probability distribution with GAN-divergence, performed better than IRGAN in most cases indicates that the generator and discriminator can be stably trained by the single-step gradient method. In addition, the pairwise method confirmed that the sampling method used in this thesis is effective for the stability of learning, especially for the MSLR-WEB30K and Yahoo datasets, suggesting the importance of pair sampling. 2) The optimal divergence differs depending on the dataset, suggesting that the number of features in the dataset affects the optimal divergence.

Chapter 6

Conclusions and Future Work

In this chapter, we first summarize the key contributions of this thesis, then describe the some interesting directions for future work.

6.1 Conclusions

In this thesis, in order to investigate the effects of different f-divergence functions on adversarial learning-to-rank, we proposed a novel adversarial training framework that applies f-divergence minimization to the training of generator and discriminator. Moreover, we adopted the single-step gradient method to smoothly update the gradient of the generator and the discriminator during training. In addition, in order for the generator to be more able to fool the discriminator when sampling document pairs, we used the Bradley-Terry model for fake document pairs and weighted sampling for ground-truth document pairs. The results of experiments with four major learning-to-rank datasets show that the proposed method outperforms the conventional adversarial learning-to-rank model. It is because the learning progresses stably by the single-step gradient method and the pair sampling method. In particular, it was suggested that the ability to sample pairs well on large datasets such as MSLR-WEB30K and Yahoo affect performance. From the analysis of the effect of each divergence on each dataset, it is inferred that the optimal divergence for learning in adversarial learning-to-rank is affected by the number of features in the dataset used for learning.

6.2 Directions for Future Work

For future work, we can consider listwise adversarial training framework that applies f-divergence minimization to the training of generator and discriminator. In the case of the listwise method, learning can be based on the total ordering relation between documents associated with the same query, rather than the single document independent of the others as in the pointwise method or the relative order between two documents as in the pairwise method. Therefore, we can expect more improvement in evaluation metrics scores.

Acknowledgement

First of all, I must express my deepest gratitude to my advisor, Dr. Yu Hai-Tao, for his guidance and patience over the years. As a student of his, I was able to receive a rigorous education with a high degree of academic freedom. I have learned a great deal about academic research, writing, and critical thinking in general from him. I am confident that these things will continue to reward me for years to come. I am also very grateful to my advisor, Dr. Joho Hideo. Throughout my graduate studies, he has provided me with a lot of help and inspiring comments. I am also grateful to my colleagues in the laboratory.

References

[1] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. pages 334–342, 2001.

[2] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st SIGIR*, pages 275–281, 1998.

[3] Tie-Yan Liu. *Learning to Rank for Information Retrieval.* Springer, 2011.

[4] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[5] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *Proceedings of TREC*, 1994.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of NeurIPS*, pages 2672–2680, 2014.

[7] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2017.

[8] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4006–4015, 2017.

[9] Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, and Nobukatsu Hojo. Stargan-vc: non-parallel many-to-many voice conversion using star generative adversarial networks. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 266–273, 2018.

[10] Takuhiro Kaneko and Hirokazu Kameoka. Cyclegan-vc: Non-parallel voice conversion using cycle-consistent adversarial networks. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2100–2104, 2018.

[11] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: a minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th SIGIR*, pages 515–524, 2017.

[12] Dae Hoon Park and Yi Chang. Adversarial sampling and training for semi-supervised information retrieval. In *Proceedings of the 28th WWW*, pages 1443–1453, 2019.

[13] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[14] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: training generative neural samplers using variational divergence minimization. In *Proceedings of 29th NIPS*, pages 271–279, 2016.

[15] Moksh Jain and Sowmya Kamath S. Proximal policy optimization for improved convergence in irgan. *ArXiv*, abs/1910.00352, 2019.

[16] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. In *Proceedings of 20th NIPS*, pages 1089–1096, 2008.

[17] Robert S. Taylor. The process of asking questions. *American Documentation*, 13(4):391–396, 1962.

[18] A. H. Lashkari, F. Mahdavi, and V. Ghomi. A boolean model in information retrieval for search engines. In *2009 International Conference on Information Management and Engineering*, pages 385–389, 2009.

[19] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[20] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *Proceedings of the 8th CIKM*, pages 316–321, 1999.

[21] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. pages 493–494, 2013.

[22] A. Agarwal, H. Raghavan, Karthik Subbian, P. Melville, R. Lawrence, David Gondek, and J. Fan. Learning to rank for robust question answering. In *Proceedings of the 21st CIKM*, 2012.

[23] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117, 1998.

[24] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems*, volume 14, pages 641–647. MIT Press, 2002.

[25] Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proceedings of NeurIPS*, pages 897–904, 2007.

[26] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*, pages 89–96, 2005.

[27] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th KDD*, pages 133–142, 2002.

[28] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th ICML*, pages 129–136, 2007.

[29] Christopher J.C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *Proceedings of NeurIPS*, pages 193–200, 2006.

[30] Tero Karras, Miika Aittala, Janne Hellsten, S. Laine, J. Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *ArXiv*, abs/2006.06676, 2020.

[31] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICML*, 2019.

[32] K. PEARSON. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can reasonably be supposed to have arisen from random sampling. *Philosophical Magazine*, 5(50):157–175, 1900.

[33] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density ratio matching under the bregman divergence: A unified framework of density ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64, 10 2011.

[34] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, 2004.

[35] Rudolf Beran. Minimum hellinger distance estimates for parametric models. *Ann. Statist.*, 5(3):445–463, 1977.

[36] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. Adversarial personalized ranking for recommendation. In *Proceedings of SIGIR*, pages 355–364, 2018.

[37] Bokun Wang, Yang Yang, Xing Xu, Alan Hanjalic, and Heng Tao Shen. Adversarial cross-modal retrieval. In *Proceedings of International Conference on Multimedia*, page 154–162, 2017.

[38] Zitai Wang, Qianqian Xu, Ke Ma, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang. Adversarial preference learning with pairwise comparisons. In *Proceedings of International Conference on Multimedia*, page 656–664, 2019.

[39] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[40] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: i. the method of paired comparisons. In *Biometrika*, pages 324–345, 1952.

[41] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: a benchmark collection for research on learning to rank for information retrieval. *Information Retrieval Journal*, 13(4):346–374, 2010.

[42] Tao Qin and Tie-Yan Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013.

[43] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the 2010 International Conference on YLRC*, pages 1–24, 2010.

[44] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[45] Diederik P. Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *ICLR*, 2015.