Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Forschungszentrum L3S
Fachgebiet Künstliche Intelligenz

Interpret Ranking Models via Set-wise Feature Selection

Masterarbeit

im Studiengang Informatik

von

Wei Zhang

Prüfer: Prof. Dr. Avishek Anand
Zweitprüfer: Prof. Dr. techn. Dipl.-Ing. Wolfgang Nejdl
Betreuer: Prof. Dr. Avishek Anand

Hannover, 30.03.2021

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 30.03.2021

Wei Zhang

Abstract

Interpret Ranking Models via Set-wise Feature Selection

Interpretability is a key requirement in algorithmic decision making today. Search engines have recently come under scrutiny for their opaqueness and biased result presentation. In this thesis I tackle the problem of interpreting ranking algorithms which are crucial in prioritizing information shown to users. Ranking algorithms take a set of data points as input, usually retrieved documents for a query, and return an ordered list of the input items. Current interpretability approaches explain single instance by selecting relevant features with respect to the output rather than a set of inputs which is a unique concern in ranking and search engines.

I propose a novel set-wise feature selection algorithm that maximizes similarity of features selected for a given set of input instances in one query. By maximizing the similarity between features selected for the retrieved input set, my approach produces a consistent explanation that can be used to explain the rank order of the inputs rather than the relevance of a single input/document to the query. Through empirical evaluation, I show that my approach is able to improve the consistency of features selected without affecting the performance of the ranking model.

Contents

1 Introduction **1**
- 1.1 Interpret Ranking model . . . 1
- 1.2 Thesis Structure . . . 2

2 Related work **3**
- 2.1 Basics of Model Interpretation . . . 3
  - 2.1.1 Interpretable Models . . . 3
  - 2.1.2 Model-Agnostic Methods . . . 4
  - 2.1.3 Example-Based Explanations . . . 4
- 2.2 Methods of Model-Agnostic Interpretation . . . 4

3 Fundamentals **7**
- 3.1 LTR Algorithms . . . 7
  - 3.1.1 Pointwise Method . . . 7
  - 3.1.2 Pairwise Method . . . 7
  - 3.1.3 Listwise Method . . . 8
- 3.2 INVASE . . . 8
  - 3.2.1 Loss Function of Selector Network . . . 8
  - 3.2.2 Back-propagation in INVASE . . . 10
  - 3.2.3 Update Predictor Network . . . 10
  - 3.2.4 Update Baseline Network . . . 10
  - 3.2.5 Update Selector Network . . . 11

4 Problem **13**
- 4.1 Problem Description . . . 13
- 4.2 Problem Formulation . . . 14

5 Proposed Approach **15**
- 5.1 Margin Loss . . . 15
- 5.2 Loss Function . . . 16
- 5.3 Framework of Pairwise INVASE Model with Margin Loss . . . 19
- 5.4 Algorithm of Pairwise INVASE Model with Margin Loss . . . 20

6 Experimental Setup **23**
6.1 Datasets 23
6.2 Models 23
6.2.1 Pairwise INVASE Model with Margin Loss 23
6.2.2 Baseline Models 24
6.3 Metrics 24
6.3.1 Jaccard Similarity 24
6.3.2 NDCG Score 25

7 Results and Discussions **27**
7.1 Performance of the pairwise INVASE model with margin loss 29
7.1.1 pairwise/pointwise INVASE model with/without margin loss 29
7.1.2 performance of the set-wise selection 31
7.1.3 pairwise INVASE model with margin loss applied on a bigger dataset 32
7.2 The effect of the number of the negative documents 33

8 Conclusion **35**

9 Appendix **37**
9.1 Compared with 7.1.2: direct intersection as set-wise selection 37

Chapter 1

Introduction

1.1 Interpret Ranking model

The search engine becomes more and more important in the last decade, in some sense we can say that it totally changes our life. If we want to buy something, we will search for it online. If there is something we don't know, we will look for help online. Everything starts with a query, which we provide for the search engine. A query represents what we want, and the search engine provides us with different relevant results. Learning-to-rank(LTR) algorithm is a bunch of methods that are used to rank results according to the query, the search engine shows us the ranked results. We want to know why the search engine ranks some results high, or some results low. Through the LTR algorithm, we can get a trained LTR model. If we know how an LTR model ranks the results, we could find the reason we want. An LTR model is usually a neural network, and one input with features for the LTR model represents one result. If we can find out a subset of features, which are the most important for the prediction, we can say that the model rank one document based on this subset of features, which is the interpretation we want. From the structure of one neural network, it is hard to tell which features are important for the prediction. One way to find out the interpretation is to use a surrogate model, we could train a surrogate model, which takes a subset of features as input and the prediction of the original LTR model as the label, to approach the performance of the original LTR model. This is the basic idea to find out the important features, or interpretation. There are different ways to do feature-selection. Globally, we can find out a subset of features that is important for all instances. And also we can find out the different important subset of features for each instance. It is reasonable to believe that the predictions of different instances are based on a different subset of features. So we focus on the instance-wise feature selection. Different from the typical instance-wise selection, we expand it to the set-wise feature selection. This idea comes from the way that we evaluate

a trained LTR model. A standard way to do evaluation is to calculate the NDCG score for each query-documents, here we use documents representing the results that the search engine returns. The calculation of the NDCG score is based on query-level, just between the instance-level and global-level. The set-wise selection means that the selections for different documents for one query are the same, but the selections between different queries are different. In datasets such as MQ2008, there are true labels representing relevance scores for each document, the way to judge one document relevant or not is based on the query, because of the relationship between query and document, there are different standards used to judge for different queries. The predictions of documents for one query should be relevant to this query, so the set-wise selection is a reasonable way to interpret a trained LTR model.

1.2 Thesis Structure

This master thesis is structured as follows. In chapter 2 the related work is presented, in which the methods about model interpretation are covered. The fundamentals related to the thesis are introduced in chapter 3. In chapter 4 the problem is formalized and the proposed method is introduced in chapter 5. The experiment details are described in chapter 6, and results and analysis are in the following chapter 7. At the end of the thesis, there is the conclusion and the future work.

Chapter 2

Related work

2.1 Basics of Model Interpretation

Machine learning techniques have been widely used in different fields, and it is usually the case that a complex model has a better performance than a simple model. Sometimes we pay much attention to the final result, such as higher accuracy or lower error, we ignore the inner process of the model. We are not clear about the way how the model makes the prediction. Sometimes we want to know why the model makes such a prediction, particularly in the medical area, the prediction is directly related to the patient's health. When the prediction made by the model has a big impact on our life, we must know the reason behind it. A recent example is the fast-developing self-driving technique, it is unbelievable to leave people's safety on a model in the car if we don't know how it works. It is necessary to dive into the model, to see the logic of it. This part will cover the basic concepts related to the model interpretation from [10].

2.1.1 Interpretable Models

The reason why a model is hard to be interpreted is related to the choice of model, which is used to solve a specific task. Some models are self-interpretable, such as the linear regression model, logistic regression model, and decision tree model. The interpretability of these models comes from the way they are trained.
An example is a linear model that can be used to model the dependence of a regression target and input features. The final trained model learns a linear relationship that includes weights for the corresponding features. One advantage of the linear regression model is linearity, it makes the prediction process simple and easy to understand. If we have an instance with numerical features, we could interpret the prediction of it in this way: increasing the numerical feature by one unit changes the estimated outcome by the corresponding weight.

2.1.2 Model-Agnostic Methods

The interpretable models are too simple, so in many tasks, they are not good enough to be adopted. Many complex models have very good performance but are hard to understand the prediction through the inner structure. A model-agnostic method can generate the interpretation without getting access to the model structure. It has a big usage since we can use it on any trained models.
An example is LIME[11] which belongs to model-agnostic methods, it can provide local interpretation for each instance. The idea behind LIME is very intuitive. Firstly we select one instance of interest for which we want to have an explanation of its black box prediction. Then we perturb the original data and get the black box predictions for the new points, and weigh the new samples according to their proximity to the instance of interest. Finally, we can train an interpretable model, such as a linear model, on the dataset with variations. We can explain the prediction by interpreting the local model.

2.1.3 Example-Based Explanations

Example-Based Explanations use some specific instances to interpret the predictions of other instances. Example-based explanations are mostly model-agnostic because they make any machine learning model more interpretable. Different from the model-agnostic methods example-based method selects instances of the dataset instead of operating on features. The idea behind it is that if A is similar to B and A caused Y, so we can think B will cause Y as well.
An example is using counterfactual explanation to interpret prediction. When the trained model is taken as a black box, we think there is a causal relationship between input and output. We can change the feature values of an instance before making the predictions and we analyze how the prediction changes. The new instance can be used as a counterfactual explanation.

2.2 Methods of Model-Agnostic Interpretation

There are some neural networks such as attention mechanism have been proposed as powerful yet human-interpretable learners, but these methods are model-specific and only suited for ML experts. Many efforts are done to find model-agnostic approaches to interpretability.
We can divide model-agnostic approaches into global-wise approaches and instance-wise approaches. The existing global-wise approaches are summarized in [5]. [6] uses max-dependency min-redundancy criteria with mutual information, [3] uses multiple hypothesis testing for global variable selection. Compared with global-wise approaches the instance-wise approaches are capable of learning instance-specific relevance. An example of the instance-

wise approach is LIME introduced in [11], it trains a local linear model for each instance, and gets the importance scores for all features of the corresponding instance. The linear model is trained in the neighborhood of the instance, the instances in the neighborhood are generated by perturbing the original instance. We can get a subset of features as the interpretation according to the weights of features for each instance. In [9] the author introduced the Shapely values from game theory to quantify the importance of features of a given input.

Gradients play an important role in the model interpretation. Since the back-propagation of the neural network is based on the gradients, we can easily get the gradients of the model output for the features of the model input. The gradients of the features reflect the importance scores of features, we can get a saliency map to mask input through the gradients from [12]. Another example is in the graph neural networks, the method ExplaiNE in [7] is proposed to provide counterfactual explanations for Network-Embedding-based link prediction methods. ExplaiNE is a mathematically principled approach that doesn't need training. The idea behind ExplaiNE is to identify the nodes which have a positive effect on the predicted link probability between a pair of nodes, and the nodes which have a negative effect. The main job of ExplaiNE is to calculate the gradients of the predicted link probability for different links. In this way the nodes with positive effects can be found as the interpretation.

Introducing the information theory into model interpretation is a conceptually different perspective from existing approaches such as the aforementioned gradients. A pair of examples are L2X[4] and INVASE[16]. L2X tries to maximize a lower bound of the mutual information between the target and the selected input features, INVASE tries to minimize the KL divergence between the target based on the original features and the target based on the selected input features. There is also an approach called GNNExplainer[15] proposed to interpret GNN models. The idea behind GNNExplainer is to maximize the mutual information between prediction and (sub-graph, subset of node features in sub-graph). There is always a surrogate model to be trained to generate interpretations based on the input and the output of the trained model.

It is usually the case that neural network is trained in an end-to-end manner, but the interpretation requires a subset of features that can be found by such as sampling, the way to get a subset of features usually breaks the process of back-propagation. INVASE and L2X adopt two different strategies to handle this issue. INVASE uses an actor-critic strategy to bypass back-propagation through the sampling and uses the predictor network to provide a reward to the selector network. Our work is mainly based on INVASE. L2X uses a generation of Gumbel-softmax trick to discretize the continuous outputs of neural network. The Gumbel-softmax makes the back propagation successful by moving the sampling operation of one variable used in back-propagation

to another variable, which is not used in the back-propagation. Concrete autoencoders[1] uses a similar strategy as L2X but generates a global feature selection. The purpose of using Gumbel-softmax is to approximate the mask that will be used to select features of the input.
There are some existing methods that interpret the ranking models. The method in [14] is proposed to generate the set-wise explanation for LTR models. It starts with an empty feature set, and iteratively adds a feature that has the largest effect on the concordant pairs from ranked results for one query. In this way, an explanation with the predefined size is generated. The author in [13] proposes a method that attempts to locally approximate a complex ranker by using a simple interpretable ranking model. The main idea is to extract the intent terms from documents that represent the information need of the user, and a simple term based ranker can be trained that maximizes coverage of preference pairs in the target ranking.
Our work is based on the INVASE, and by introducing the margin loss to implement the set-wise selection for LTR models.

Chapter 3

Fundamentals

3.1 LTR Algorithms

This part will cover the basic ideas of different LTR algorithms from [8]. LTR is widely used in search, recommendation, and advertising. In the machine learning field, we can take the LTR as a supervised learning task. There are three different types of LTR: pointwise, pairwise, listwise. The main difference among them is how they organize the training data from the original data.

3.1.1 Pointwise Method

A typical data format in LTR is (query id, relevance score, document features), A pointwise model focus to learn the relationship between document and relevance score. The input of the pointwise model is features of one document, and the output is the predicted relevance score. Since we know the true relevance score, the pointwise model can be trained as a classification task, regression task, or ordinary regression task.
The pointwise model is the easiest way to construct an LTR model, however, it has multiple drawbacks. An LTR model focus on the final ranked results given one query, which means that the relative relevance scores among documents are more important. And the pointwise model doesn't consider the relationship between query and document. Different queries usually have different numbers of documents in the original data, if we use a pointwise model to learn a global score function, the loss will be dominated by the queries with much more documents.

3.1.2 Pairwise Method

For a search system, after receiving a user query, the system returns a list of related documents, so the key to the problem is to determine the order relationship between the documents. The pointwise method considers the

problem from the perspective of the classification score of a single document and does not consider the order relationship between the documents. The pairwise method transforms the sorting problem into a sorting problem of multiple pairs and compares the order of different documents. Different from the pointwise model, the input of the pairwise model is a pair of documents that are from one query. Given one query, it is important to find out the order relationship between a pair of documents (d_1, d_2), if d_1 has a higher relevance score, we call it a positive pair, otherwise negative pair. It is called the pairwise method because the training process and training goal of this machine learning method is to determine whether a document pair is positive.
There are still some drawbacks of the pairwise method. The pairwise method considers the relative order of the two documents but does not consider the position of the document in the ranked list. The document ranked higher in the list is more important. If the document at the top has a judgment error, the cost is significantly higher than other documents. Different queries have large differences in the number of related documents, so after converting to document pairs, some queries can have hundreds of corresponding document pairs, while some queries have only a dozen corresponding document pairs.

3.1.3 Listwise Method

The pointwise method treats each document in the training set as a training instance, and the pairwise method uses any two documents of the same query as a training instance. Different from the pointwise and pairwise methods, the listwise method organizes the training data differently. It uses all documents corresponding to each query as a training instance, which is why it is called the listwise method. The listwise method directly considers the overall documents of one query and optimizes the ranking evaluation index, such as commonly used MAP, NDCG.

3.2 INVASE

In this part, we will introduce the INVASE[16]. As we can see from the INVASE framework in Figure 3.1, what the INVASE does is to make the performance of the predictor network using the masked input approach the performance of the baseline network using original input. In the following, we mainly introduce the loss definition and the update of INVASE in training.

3.2.1 Loss Function of Selector Network

The KL divergence is used to formalize the difference between the predictor output and the baseline output. $Y|\mathbf{X}$ denotes the conditional distribution of a random variable Y given a random variable $\mathbf{X}$. Since the INVASE focuses

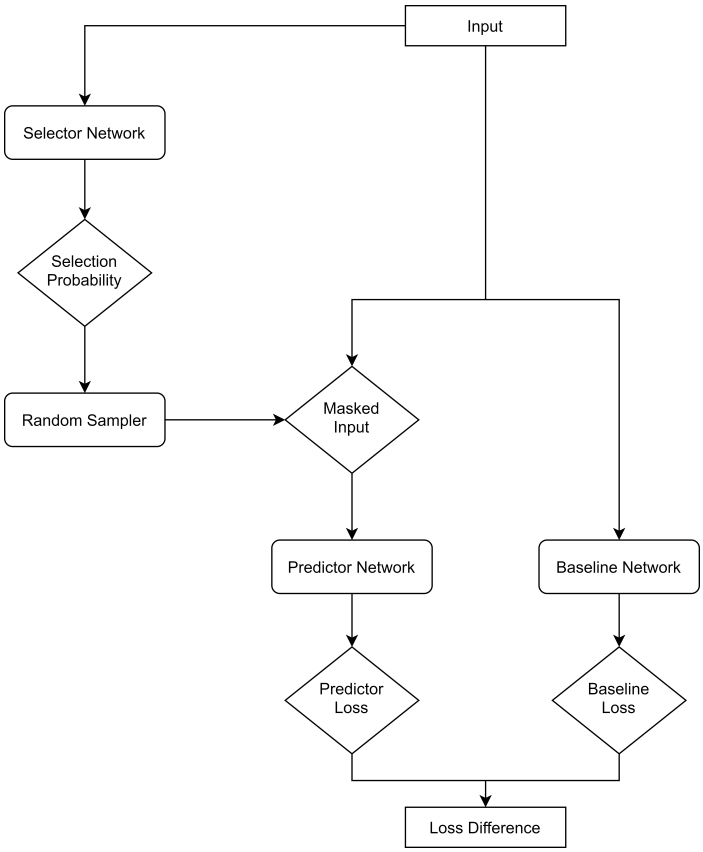

Figure 3.1: framework of INVASE

on the interpretation of one instance, the KL divergence is used with respect to the realization of $\mathbf{X}$, denoted as $\mathbf{x}$. We use $S(\mathbf{x})$ to denote the mask generated by the selector network for $\mathbf{x}$, so the loss function of the selector network of INVASE is defined as:

$$\mathcal{L}(S) = \mathbb{E}_{\mathbf{x}\sim p_X}\left[KL\left(Y|\mathbf{X}=\mathbf{x}\|Y|\mathbf{X}^{(S(\mathbf{X}))}=\mathbf{x}^{(S(\mathbf{x}))}\right)+\lambda\|S(\mathbf{x})\|\right] \tag{3.1}$$

where $\|\cdot\|$ simply denotes the number of non-zero entries of a vector and λ is used to control the number of the selected features.

$$\begin{aligned} &KL\left(Y|\mathbf{X}=\mathbf{x}\|Y|\mathbf{X}^{(S(\mathbf{X}))}=\mathbf{x}^{(S(\mathbf{x}))}\right) \\ &\quad=\mathbb{E}_{y\sim Y|\mathbf{X}=\mathbf{x}}\left[\log\left(\frac{p_Y(y\mid\mathbf{x})}{p_Y(y\mid\mathbf{x}^{(S(\mathbf{x}))})}\right)\right] \\ &\quad=\mathbb{E}_{y\sim Y|\mathbf{X}=\mathbf{x}}\left[\log\left(p_Y(y\mid\mathbf{x})\right)-\log\left(p_Y\left(y\mid\mathbf{x}^{(S(\mathbf{x}))}\right)\right)\right] \\ &\quad=\int_{\mathcal{Y}} p_Y(y\mid\mathbf{x})\left[\log\left(p_Y(y\mid\mathbf{x})\right)-\log\left(p_Y\left(y\mid\mathbf{x}^{(S(\mathbf{x}))}\right)\right)\right]dy \end{aligned} \tag{3.2}$$

where $p_Y(\cdot \mid \cdot)$ denotes the appropriate conditional densities of Y . Define

$$l(\mathbf{x}, \mathbf{s}) = \int_{\mathcal{Y}} p_Y(y \mid \mathbf{x}) \left[\log \left(p_Y(y \mid \mathbf{x}) \right) - \log \left(p_Y \left(y \mid \mathbf{x}^{(s)} \right) \right) \right] dy \tag{3.3}$$

so the final loss can be written as

$$\mathcal{L}(S) = \mathbb{E}_{\mathbf{x} \sim p_X} [l(\mathbf{x}, S(\mathbf{x})) + \lambda \| S(\mathbf{x}) \|] \tag{3.4}$$

where $\| \cdot \|$ denotes l_0 (pseudo-)norm.

3.2.2 Back-propagation in INVASE

A naive way to think of the training INVASE is end-to-end, but from the framework of INVASE, we can find the sampling operation on the output of the selector part, due to sampling it is impossible to do back-propagation since sampling operation is not differentiable. By looking at the way of calculating gradients of selector loss we can find that we can update selector network without considering predictor network and baseline network, also the same case for updating predictor and baseline network. It will be shown in the following part. By updating these three sub-networks separately the INVASE models avoid passing gradients through the sampling process. It is called tricky because how to deal with the sampling process during training is the key point of other models, such as L2X[4].

3.2.3 Update Predictor Network

To approximate the densities in (3.3), define a pair of functions $f^\phi : \mathcal{X}^* \times \{0,1\}^d \to [0,1]^c$ parametrized by ϕ and $f^\gamma : \mathcal{X} \to [0,1]^c$ parametrized by γ that will estimate $p_Y \left(\cdot \mid \mathbf{x}^{(S(\mathbf{x}))} \right)$ and $p_Y(\cdot \mid \mathbf{x})$ respectively.
For the predictor network, in the case of classification the following loss is used

$$l_1(\phi) = -\mathbb{E}_{(\mathbf{x},y) \sim p, \mathbf{s} \sim \pi_\theta(\mathbf{x}, \cdot)} \left[\sum_{i=1}^{c} y_i \log \left(f_i^\phi \left(\mathbf{x}^{(\mathbf{s})}, \mathbf{s} \right) \right) \right] \tag{3.5}$$

where y_i is the i th component of the one-hot encoding of y and π_θ is the distribution induced by selector network which will be defined in the following part. f^ϕ is implemented as a fully connected neural network, CNNs and RNNs are also possible when appropriate.

3.2.4 Update Baseline Network

It is similar to define the loss for baseline network, but without mask:

$$l_3(\gamma) = -\mathbb{E}_{(\mathbf{x},y) \sim p} \left[\sum_{i=1}^{c} y_i \log \left(f_i^\gamma(\mathbf{x}) \right) \right] \tag{3.6}$$

3.2.5 Update Selector Network

For fixed ϕ, γ we could define loss estimator, $\hat{l}$, by

$$\hat{l}(\mathbf{x}, \mathbf{s}) = -\left[\sum_{i=1}^{c} y_i \log\left(f_i^{\phi}\left(\mathbf{x}^{(\mathbf{s})}, \mathbf{s}\right)\right) - \sum_{i=1}^{c} y_i \log\left(f_i^{\gamma}(\mathbf{x})\right)\right] \tag{3.7}$$

The selector network induces a probability distribution over the selection space $\left(\{0,1\}^d\right)$, with the probability of a given joint selection vector $\mathbf{s} \in \{0,1\}^d$ being given by

$$\pi_\theta(\mathbf{x}, \mathbf{s}) = \Pi_{i=1}^{d} \hat{S}_i^{\theta}(\mathbf{x})^{s_i}\left(1 - \hat{S}_i^{\theta}(\mathbf{x})\right)^{1-s_i} \tag{3.8}$$

where the selector function $S : \mathcal{X} \to \{0,1\}^d$ is realized by using a single neural network, $\hat{S}^{\theta} : \mathcal{X} \to [0,1]^d$ parameterized by weights θ, that outputs a probability for selecting each feature (i.e. the ith component of $\hat{S}^{\theta}(\mathbf{x})$ will denote the probability with which we select the ith feature).
Through (3.4) and (3.7) we can define the following loss for selector network

$$\begin{aligned} l_2(\theta) &= \mathbb{E}_{(\mathbf{x},y)\sim p}\left[\mathbb{E}_{\mathbf{s}\sim\pi_\theta(\mathbf{x},\cdot)}\left[\hat{l}(\mathbf{x},\mathbf{s}) + \lambda\|\mathbf{s}\|_0\right]\right] \\ &= \int_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x}, y)\left(\sum_{\mathbf{s}\in\{0,1\}^d} \pi_\theta(\mathbf{x},\mathbf{s})\left(\hat{l}(\mathbf{x},\mathbf{s}) + \lambda\|\mathbf{s}\|_0\right)\right) dxdy \end{aligned} \tag{3.9}$$

Calculate the gradient of this loss with respect to θ

$$\begin{aligned} \nabla_\theta l_2(\theta) &= \int_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x}, y)\left(\sum_{\mathbf{s}\in\{0,1\}^d} \nabla_\theta\pi_\theta(\mathbf{x},\mathbf{s})\left(\hat{l}(\mathbf{x},\mathbf{s}) + \lambda\|\mathbf{s}\|_0\right)\right) dxdy \\ &= \int_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x}, y)\left(\sum_{\mathbf{s}\in\{0,1\}^d} \frac{\nabla_\theta\pi_\theta(\mathbf{x},\mathbf{s})}{\pi_\theta(\mathbf{x},\mathbf{s})}\pi_\theta(\mathbf{x},\mathbf{s})\left(\hat{l}(\mathbf{x},\mathbf{s}) + \lambda\|\mathbf{s}\|_0\right)\right) dxdy \\ &= \int_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x}, y)\left(\sum_{\mathbf{s}\in\{0,1\}^d} \nabla_\theta \log\pi_\theta(\mathbf{x},\mathbf{s})\pi_\theta(\mathbf{x},\mathbf{s})\left(\hat{l}(\mathbf{x},\mathbf{s}) + \lambda\|\mathbf{s}\|_0\right)\right) dxdy \\ &= \mathbb{E}_{(\mathbf{x},y)\sim p}\left[\mathbb{E}_{\mathbf{s}\sim\pi_\theta(\mathbf{x},\cdot)}\left[\left(\hat{l}(\mathbf{x},\mathbf{s}) + \lambda\|\mathbf{s}\|_0\right)\nabla_\theta \log\pi_\theta(\mathbf{x},\mathbf{s})\right]\right] \end{aligned} \tag{3.10}$$

As you can see, updating the selector network can be done alone.

Chapter 4

Problem

4.1 Problem Description

There are different kinds of ways to interpret models, mostly focus on global feature selection or instance-wise feature selection. The global feature selection means that the contributions of different features are the same for predictions of different instances. And the instance-wise feature selection means that the predictions of different instances are affected by different features. If we want to interpret trained LTR models, we can choose global feature selection or instance-wise feature selection. The instance-wise interpretation usually provides us with a more accurate way to understand why the trained model makes such a prediction on a specific instance. As we all know that such as NDCG score is a metric to get the performance of trained LTR models. Unlike usually trained models we can easily use such as accuracy to evaluate them, the NDCG score is calculated on query-documents. For each query there are some documents with corresponding true labels, we could get predictions of these documents through the trained LTR models. By comparing two rank lists, one is from true labels, another one is from predictions, we can get one NDCG score for this query-documents, and further an average NDCG score of all query-documents. We can see that the computation restricts to set data(query-documents), not one single instance or all instances. If we use the instance-wise way to interpret the trained LTR models, for each query there will be different selected features for different documents. The true labels of documents of one query are decided by a subset of features, and we think they should be the same. Based on the shared subset of features we can decide if these documents are relevant to this query. So now what we want is a set-wise interpretation for the trained LTR models. The predictions of different documents of one query should be explained the same, which means the selected features of them are the same. But for different queries, the selected features should be different, the reasonable way to think of it is that if one

document is relevant to different queries or not should be decided by different features.

4.2 Problem Formulation

Given a trained LTR model M, and the data that consists of queries Q_i, in which (x_{ij}, y_{ij}) represents the jth document with d-dimensional features and the corresponding relevance score in query Q_i that is predicted by M. We use $\mathbf{x}_{ij}$ to denote the jth document in Q_i and $\mathcal{X}$ to denote a d-dimensional feature space. Our goal is to find a selector function, $S : \mathbf{x}_{ij} \in \mathcal{X} \rightarrow \{0,1\}^d$, such that

$$\mathbf{P}\left(Y_{ij} \mid \mathbf{X}_{ij}^{(S(\mathbf{x}_{ij}))} = \mathbf{x}_{ij}^{(S(\mathbf{x}_{ij}))}\right) = \mathbf{P}(Y_{ij} \mid \mathbf{X}_{ij} = \mathbf{x}_{ij}) \tag{4.1}$$

s.t. $S(\mathbf{x}_{ij}) = S(\mathbf{x}_{ik})$, $\forall$ $\mathbf{x}_{ij}$, $\mathbf{x}_{ik}$ from Q_i, and $S(\mathbf{x})$ is minimal.

Chapter 5

Proposed Approach

The proposed method is based on INVASE[16]: instance-wise variable selection using neural networks. The basic idea of it is to minimize the KL divergence between the conditional distributions $Y|X$ and $Y|X_S$. In this part, I will introduce how to expand it to realize the set-wise interpretation.

5.1 Margin Loss

Unlike other losses, such as cross-entropy and mean squared error which are designed to make the model learn to predict directly, the purpose of the margin loss is to learn relative distances among input instances. The way to measure the distance between a pair of instances we used is cross-entropy. In the task of identifying if two faces belong to one person, a convolutional neural network can be trained to learn a new representation for each instance. A well-trained model can generate very similar representations for the input faces that belong to one person, and generate very different representations for the input faces of different persons. There are two different ways to use margin loss, one is to calculate margin loss by using a pair of instances, another one is to use three instances. In the first case, the formula is:

$$L = \begin{cases} \mathrm{d}\left(r_a, r_p\right) & \text{positive pair } \left(x_a, x_p\right) \\ \max\left(0, m - \mathrm{d}\left(r_a, r_n\right)\right) & \text{negative pair } \left(x_a, x_n\right) \end{cases} \tag{5.1}$$

A positive pair (x_a, x_p) consists of one anchor instance and a positive instance which has the same label as the same anchor instance, in the aforementioned examples two faces from one person can form a positive pair, a negative pair (x_a, x_n) consists of two faces which are from different persons. r is the representation that the model learns for the instance. In the following way, the margin loss with two instances can be used directly as a loss. y is a binary variable, $y = 1$ when (r_0, r_1) is the representations of a positive pair, $y = 0$ when (r_0, r_1) is the representations of a negative pair,

$$L\left(r_0, r_1, y\right) = y \left\|r_0 - r_1\right\| + (1 - y) \max\left(0, m - \left\|r_0 - r_1\right\|\right) \tag{5.2}$$

In practice, we found that the margin loss with three instances has a better performance than only using two instances. The margin loss with three instances is also called triple loss. The formula of triplet loss is:

$$L\left(r_a, r_p, r_n\right) = \max\left(0, m + \mathrm{d}\left(r_a, r_p\right) - \mathrm{d}\left(r_a, r_n\right)\right) \tag{5.3}$$

The triplets from original data can be divided into three classes, easy triplet, hard triplet and semi-hard triplet as in the following figure. The triplet is called easy triplet when $\mathrm{d}\left(r_a, r_n\right) > \mathrm{d}\left(r_a, r_p\right) + m$, called hard triplet when $\mathrm{d}\left(r_a, r_n\right) < \mathrm{d}\left(r_a, r_p\right)$, and called semi-hard triplet when $\mathrm{d}\left(r_a, r_p\right) < \mathrm{d}\left(r_a, r_n\right) < \mathrm{d}\left(r_a, r_p\right) + m$. In the case of easy triplet, compared with the distance between r_p and r_a, the distance between r_n and r_a is big enough so there is no need to update model parameters, the value of loss is 0. In the case of hard triplet, r_n is much closer to r_a than r_p, the value of loss is positive and bigger than margin m. In the case of semi-hard triplet r_p is closer to r_a than r_n, but r_n is not far enough from r_a, the value of loss is positive and smaller than margin m. The triplet loss focus on the hard triplets and the semi-hard triplets.

5.2 Loss Function

The goal of pairwise INVASE is to provide a set-wise explanation for trained ranking models. For better performance, we train the predictor network and the baseline network in a pairwise way. We get the mask for each instance through sampling on the output of the selector network, each element of which represents the probability of one feature that can be selected. The input of pairwise INVASE is $(\mathbf{x_1}, y_1, \mathbf{x_2}, y_2)$, $\mathbf{x}$ represents one document and y is corresponding relevance score. After feeding $\mathbf{x_1}$ and $\mathbf{x_2}$ through selector separately, we can get two different masks $\mathbf{s_1}$ and $\mathbf{s_2}$ for these two documents. The input of the predictor network is two instances and the corresponding masks, we could get a pair of documents with selected features by doing element-wise multiplication. Then we can train the predictor network as the RankNet[2]. For baseline network, we can just feed two documents into it and train it directly as the RankNet.
We can manually choose document-pair $(\mathbf{x_1}, \mathbf{x_2})$, of which $\mathbf{x_1}$ has a higher relevance score. In order to train pairwise predictor, the input is $(\mathbf{x_1}, s_1, \mathbf{x_2}, s_2)$ and the label is 1. And the pairwise predictor is used in pointwise way to do inference. f^{ϕ} is referred as predictor. The loss of predictor is

$$\begin{aligned} l_1(\phi) = -\mathbb{E}_{(\mathbf{x_1},\mathbf{x_2})\sim p, \mathbf{s_1}\sim\pi_\theta(\mathbf{x_1},\cdot), \mathbf{s_2}\sim\pi_\theta(\mathbf{x_2},\cdot)} \\ \left[\sum_{i=0}^{1} t_i \log\left(\mathbf{Sigmoid}\left(f_i^{\phi}\left(\mathbf{x_1}^{(\mathbf{s_1})}, \mathbf{s_1}\right) - f_i^{\phi}\left(\mathbf{x_2}^{(\mathbf{s_2})}, \mathbf{s_2}\right)\right)\right)\right] \end{aligned} \tag{5.4}$$

Pairwise baseline follows the same logic as above, but without the mask for instance. And the baseline is used in a pointwise way to do inference. f^{γ} is referred as the baseline. The loss of baseline is

$$l_3(\gamma) = -\mathbb{E}_{(\mathbf{x_1},\mathbf{x_2})\sim p}\left[\sum_{i=0}^{1} t_i \log\left(\mathbf{Sigmoid}\left(f_i^{\gamma}\left(\mathbf{x_1}\right) - f_i^{\gamma}\left(\mathbf{x_2}\right)\right)\right)\right] \tag{5.5}$$

Since we feed two instances into selector of pairwise INVASE firstly, we can add a constraint on the output of them, to force selector to generate the same probability distribution of them. $\mathbf{x_1}$ and $\mathbf{x_2}$ come from the same query, in the experiment we found it not enough to get what we want. It is usually the case that the selector network generates the same probability distributions for different documents from different queries, which is a global selection. This is because we only consider the relationship between documents from one query, so we choose to feed the selector network some documents from other queries, the output of them will not be fed further into predictor. In this case, we call $\mathbf{x_1}$ the anchor document, and $\mathbf{x_2}$ is the positive document that comes from the same query as $\mathbf{x_1}$, $\mathbf{x_3}$ are the negative documents that come from different queries. We use margin loss to force selector to generate probability distributions for $\mathbf{x_1}$ and $\mathbf{x_3}$ as different as possible, and generate same probability distributions for $\mathbf{x_1}$ and $\mathbf{x_2}$. The loss of selector of pairwise INVASE is

$$\begin{aligned}
l_2(\theta) &= \mathbb{E}_{((\mathbf{x_1},\mathbf{y_1}),(\mathbf{x_2},\mathbf{y_2}),(\mathbf{x_3},\mathbf{y_3}))}\left[\frac{1}{2}\left(\mathbb{E}_{\mathbf{s_1}\sim\pi_\theta(\mathbf{x_1},\cdot)}\left[\hat{l}(\mathbf{x_1},\mathbf{s_1}) + \lambda\|\mathbf{s_1}\|_0\right]\right.\right. \\
&\quad \left.\left. + \mathbb{E}_{\mathbf{s_2}\sim\pi_\theta(\mathbf{x_2},\cdot)}\left[\hat{l}(\mathbf{x_2},\mathbf{s_2}) + \lambda\|\mathbf{s_2}\|_0\right]\right) \right. \\
&\quad \left. + max\left(0, m + \mathbb{CE}\left(\hat{S}^{\theta}\left(\mathbf{x_1}\right), \hat{S}^{\theta}\left(\mathbf{x_2}\right)\right) - \mathbb{CE}\left(\hat{S}^{\theta}\left(\mathbf{x_1}\right), \hat{S}^{\theta}\left(\mathbf{x_3}\right)\right)\right)\right] \\
&= \int_{(X_1,Y_1)\times(X_2,Y_2)\times(X_3,Y_3)} p\Big((\mathbf{x_1},y_1),(\mathbf{x_2},y_2),(\mathbf{x_3},y_3)\Big)\Big[\\
&\quad \frac{1}{2}\Bigg(\sum_{\mathbf{s_1}\in\{0,1\}^d} \pi_\theta(\mathbf{x_1},\mathbf{s_1})\left(\hat{l}(\mathbf{x_1},\mathbf{s_1}) + \lambda\|\mathbf{s_1}\|_0\right) \\
&\quad + \sum_{\mathbf{s_2}\in\{0,1\}^d} \pi_\theta(\mathbf{x_2},\mathbf{s_2})\left(\hat{l}(\mathbf{x_2},\mathbf{s_2}) + \lambda\|\mathbf{s_2}\|_0\right)\Bigg) \\
&\quad + max\left(0, m + \mathbb{CE}\left(\hat{S}^{\theta}\left(\mathbf{x_1}\right), \hat{S}^{\theta}\left(\mathbf{x_2}\right)\right) - \mathbb{CE}\left(\hat{S}^{\theta}\left(\mathbf{x_1}\right), \hat{S}^{\theta}\left(\mathbf{x_3}\right)\right)\right) \\
&\quad \Big] d(x_1,y_1)d(x_2,y_2)d(x_3,y_3)
\end{aligned} \tag{5.6}$$

m is the margin value. In the experiments, we found that a smaller margin has a better performance. Compared with identifying human face example,

we can consider each query as a person, and each document for this query is considered as different faces of this person. In practice, it is obvious to distinguish different persons, so it is not hard for the model to learn different representations for different persons. However, it is not easy to learn representations for different documents from different queries, since the query information is only reflected in the relevance score. A smaller margin means that the model focuses more on the hard triplets to distinguish documents from different queries. Since the output of the selector is not a distribution of all features, we can not just use cross-entropy to measure the distance between two outputs from the selector. We feed the output of the selector into a softmax layer, then we can use cross-entropy as a metric. The cross-entropy is used as

$$\mathbb{CE}\left(\hat{S}^{\theta}\left(\mathbf{x_1}\right), \hat{S}^{\theta}\left(\mathbf{x_2}\right)\right) = -\left[\hat{S}^{\theta}(\mathbf{x_1}) \log(\hat{S}^{\theta}(\mathbf{x_2})) + (1 - \hat{S}^{\theta}(\mathbf{x_1})) \log(1 - \hat{S}^{\theta}(\mathbf{x_2}))\right] \tag{5.7}$$

5.3 Framework of Pairwise INVASE Model with Margin Loss

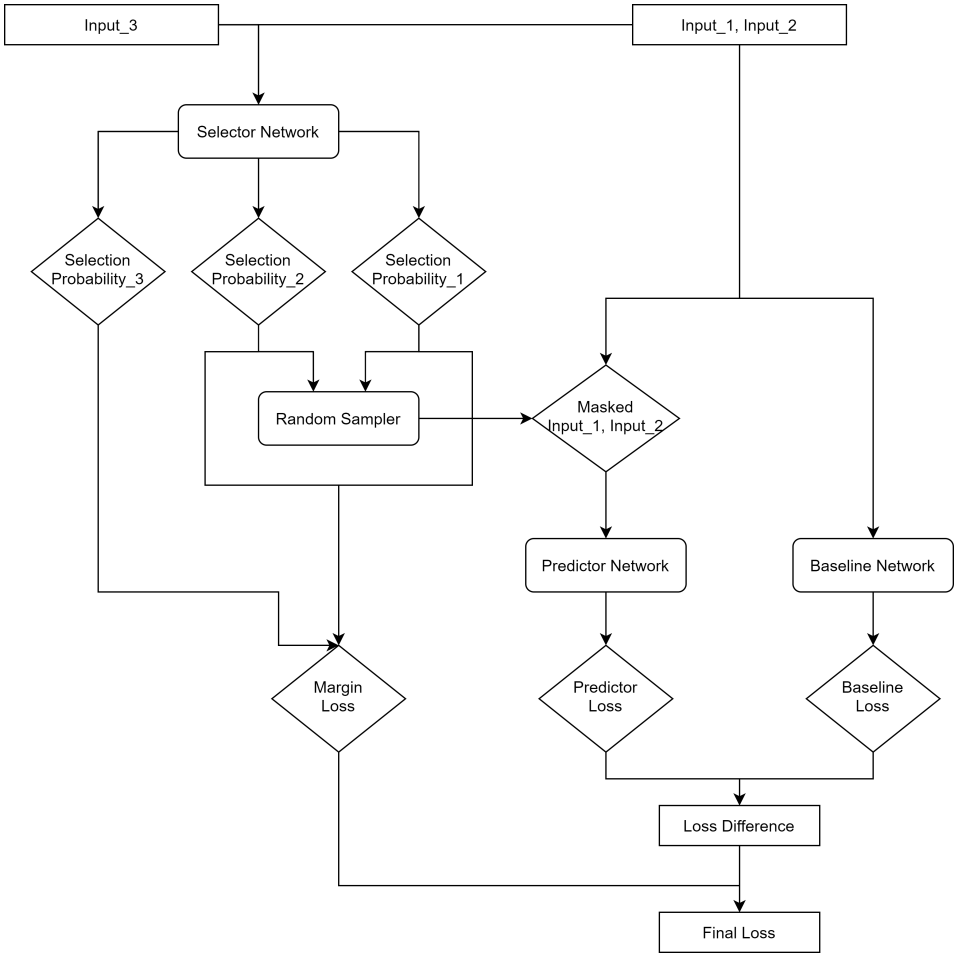

Figure 5.1: Framework of pairwise INVASE model with margin loss

5.4 Algorithm of Pairwise INVASE Model with Margin Loss

Algorithm 1 Pseudo-code of pairwise INVASE model with margin loss

Inputs: learning rate $\alpha, \beta > 0$, $n_{mb} > 0$, dataset $\mathcal{D}$. Initialize θ, ϕ, γ

Generate triplets $(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3)$ from $\mathcal{D}$ with labels (Y_1, Y_2, Y_3)

while Converge **do**

  sample a mini-batch triplets $(\mathbf{x}_{1j}, \mathbf{x}_{2j}, \mathbf{x}_{3j})_{j=1}^{n_{mb}}$ from generated triplets.

  for $j = 1, ..., n_{mb}$ **do**

    Calculate selection probabilities

$$\left(p_{11}^j, \ldots, p_{1d}^j\right) \leftarrow \hat{S}^\theta\left(\mathbf{x}_{1j}\right)$$
$$\left(p_{21}^j, \ldots, p_{2d}^j\right) \leftarrow \hat{S}^\theta\left(\mathbf{x}_{2j}\right)$$
$$\left(p_{31}^j, \ldots, p_{3d}^j\right) \leftarrow \hat{S}^\theta\left(\mathbf{x}_{3j}\right)$$

    for $i = 1, ..., d$ **do**

$$s_{1i}^j \sim \text{Ber}\left(p_{1i}^j\right); s_{2i}^j \sim \text{Ber}\left(p_{2i}^j\right);$$

    Calculate loss difference

$$\hat{l}_j\left(\mathbf{x}_{1j}, \mathbf{s}_{1j}\right) \leftarrow -\left[\sum_{i=1}^{c} y_{1i}^j \log\left(f_i^\phi\left(\mathbf{x}_{1j}^{(\mathbf{s}_{1j})}, \mathbf{s}_{1j}\right)\right) - \sum_{i=1}^{c} y_{1i}^j \log\left(f_i^\gamma\left(\mathbf{x}_{1j}\right)\right)\right]$$

$$\hat{l}_j\left(\mathbf{x}_{2j}, \mathbf{s}_{2j}\right) \leftarrow -\left[\sum_{i=1}^{c} y_{2i}^j \log\left(f_i^\phi\left(\mathbf{x}_{2j}^{(\mathbf{s}_{2j})}, \mathbf{s}_{2j}\right)\right) - \sum_{i=1}^{c} y_{2i}^j \log\left(f_i^\gamma\left(\mathbf{x}_{2j}\right)\right)\right]$$

  update the selector network parameters θ

$$\mathbf{Gradient_{1j}} = \left(\hat{l}_j\left(\mathbf{x}_{1j}, \mathbf{s}_{1j}\right) + \lambda \left\|\mathbf{s}_{1j}\right\|\right) \nabla_\theta \log \pi_\theta\left(\mathbf{x}_{1j}, \mathbf{s}_{1j}\right)$$

$$\mathbf{Gradient_{2j}} = \left(\hat{l}_j\left(\mathbf{x}_{2j}, \mathbf{s}_{2j}\right) + \lambda \left\|\mathbf{s}_{2j}\right\|\right) \nabla_\theta \log \pi_\theta\left(\mathbf{x}_{2j}, \mathbf{s}_{2j}\right)$$

$$\mathbf{Gradient_{3j}} = \nabla_\theta \mathbf{MarginLoss}\left(\hat{S}^\theta\left(\mathbf{x}_{1j}\right), \hat{S}^\theta\left(\mathbf{x}_{2j}\right), \hat{S}^\theta\left(\mathbf{x}_{3j}\right)\right)$$

$$\theta \leftarrow \theta - \alpha \frac{1}{n_{mb}} \sum_{j=1}^{n_{mb}} \left(\frac{1}{2}\left(\mathbf{Gradient_{1j}} + \mathbf{Gradient_{2j}}\right) + \mathbf{Gradient_{3j}}\right)$$

10: update the predictor network parameters ϕ

$$\phi \leftarrow \phi - \beta \frac{1}{n_{mb}} \sum_{j=1}^{n_{mb}} \sum_{i=0}^{1} t_i \times \nabla_{\phi} \left[\log \left(\mathbf{Sigmoid} \left(f_i^{\phi} \left(\mathbf{x_{1j}}^{(\mathbf{s_{1j}})}, \mathbf{s_{1j}} \right) - f_i^{\phi} \left(\mathbf{x_{2j}}^{(\mathbf{s_{2j}})}, \mathbf{s_{2j}} \right) \right) \right) \right]$$

11: update the baseline network parameters γ

$$\gamma \leftarrow \gamma - \beta \frac{1}{n_{mb}} \sum_{j=1}^{n_{mb}} \sum_{i=0}^{1} t_i \times \nabla_{\gamma} \left[\log \left(\mathbf{Sigmoid} \left(f_i^{\gamma} \left(\mathbf{x_{1j}} \right) - f_i^{\gamma} \left(\mathbf{x_{2j}} \right) \right) \right) \right]$$

Chapter 6

Experimental Setup

6.1 Datasets

The datasets we used are MQ2008 and MSLRWEB-10K from LETOR[1], which is a package of benchmark datasets for research on Learning to Rank algorithms. Both data sets have the same format, they contain labeled documents with the corresponding query-id. MQ2008 has 46 features and MSLRWEB-10K is a bigger dataset with 136 features. Each instance of both datasets has one query-id, the corresponding features representing one document, and one relevance score indicating the relationship between the query and the document. According to the 5 folder splitting, the dataset is divided into 3:1:1 as training data, validation data, and test data. For the five folders of both datasets, the different folders only differ in the order, so for all experiments, we use the fold1 data. We take the true relevance score in the dataset as the prediction of a trained model, so the experiments can be seen as interpreting a ranking model.

6.2 Models

6.2.1 Pairwise INVASE Model with Margin Loss

The selector network is implemented as a fully-connected neural network with 10 layers. Since we want the performance of the predictor network to approach the performance of the baseline, so we need to keep the structure of them the same to find out a subset of features representing the whole instance. The implementations of the baseline network and the predictor are the same, which are also fully-connected neural networks with 3 layers. They only differ in the input, the original instance used for the baseline network, the masked instance used for the predictor network. Each layer of

[1] https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/!letor-4-0

the selector network has 300 hidden units, and there are 200 hidden units for each layer of the predictor network and baseline network. The batch size we used is 128. During experiments we found that the learning rate for optimizer of selector network has a crucial effect on the performance of the model, the best value we tried for now is 1e-6. The learning rates for optimizers of the predictor network and baseline network are the same with value 1e-4. Adam optimizers are used for these three networks.

6.2.2 Baseline Models

- pairwise INVASE model without margin loss
  By removing the margin loss part, we trained this baseline to examine the effect of the margin loss on the performance of the pairwise INVASE model.

- pointwise INVASE model with margin loss
  The pointwise INVASE model trains the predictor network and the baseline network in a pointwise way. Using margin loss could lead to a set-wise selection.

- pointwise INVASE model without margin loss
  The pointwise INVASE model without margin loss is just the original INVASE model.

- L2X model
  We use the L2X model to generate the set-wise selection as a baseline. L2X model follows the selector-predictor framework, so we use the same network setting as the selector and the predictor in pairwise INVASE.

6.3 Metrics

6.3.1 Jaccard Similarity

We choose Jaccard Similarity to evaluate the generated set-wise selection. The Jaccard Similarity is used to measure similarity between finite sample sets and is defined as :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6.1}$$

After we get a trained model, we only use the selector network of it, for each document in one query the selector network generates the corresponding probabilities of features, we choose the features with probabilities bigger than 0.5. We calculate an average Jaccard Similarity for every pair of selected features of two documents in one query. We also calculate an average Jaccard Similarity for sampled pair of selected features of two documents from different queries. An ideal result for the set-wise selection is that

the average Jaccard Similarity in one query is very high, which means the selected features for documents in one query are similar, and the average Jaccard Similarity from different queries is very low, which means the selected features from different queries are different.

6.3.2 NDCG Score

NDCG, Normalized Discounted Cumulative Gain, is used to evaluate the L2R algorithm and developed from DCG. There are two factors considered to do the evaluation. The first factor is that the document with a higher relevance score influences more the result of the evaluation than the document with a lower relevance score. The second factor is that the result of the evaluation is positively related to the position of documents with higher relevance scores. The calculation of DCG is as following :

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{\mathrm{rd}_i} - 1}{\log_2(i+1)} \tag{6.2}$$

i is the position in the ranking list, $\mathrm{rel_i}$ represents the relevance score of document in position i. Since there are different numbers of ranked documents for each query and DCG reflects a cumulative influence, we need to normalize it to compare two ranked documents. IDCG is the ideal score of ranked documents.

$$\mathrm{IDCG_p} = \sum_{i=1}^{|REL|} \frac{2^{\mathrm{rel}} - 1}{\log_2(i+1)} \tag{6.3}$$

$$\mathrm{NDCG_p} = \frac{DCG_p}{IDCG_p} \tag{6.4}$$

Chapter 7

Results and Discussions

There are three hyper-parameters tuned in the experiments: the lambda value used to control the number of selected features per instance, the margin value used to force the selector to generate the set-wise selection, and the number of negative documents used in margin loss.
The following research questions are explored:

- Q1: Compared with the baselines, can we get the best performance by using the pairwise INVASE model with margin loss?

- Q2: Does the number of negative documents affect the performance?

In section 7.1, firstly we compare the performance of these models: pairwise INVASE model with margin loss, pairwise INVASE model without margin loss, pointwise INVASE model with margin loss, and pointwise INVASE model without margin loss. Secondly, since the trained selector network can not always generate the same selected features for each instance in one query, we need to get a common mask for one query which will be taken as the final set-wise interpretation.
And also we checked the performance of the pairwise INVASE model with margin loss on a bigger dataset MSLRWEB-10k.
In section 7.2 the effect of using a different number of negative documents was examined.
The terms used in the results:

- JS indicates the Jaccard Similarity calculated on the selected features with probabilities higher than 0.5

- P@K indicates the calculated NDCG@K score of the predictor by using features with probabilities higher than 0.5

- B@K indicates the calculated NDCG@K score of the baseline by using features with probabilities higher than 0.5

7.1 Performance of the pairwise INVASE model with margin loss

In MQ2008 each document has 46 features. The number of negative samples used in one instance in training is set to 5. Since the selector network could generate the different number of the selected features for each instance when we tested the model, we used the average number to denote the size of the selection per instance.

7.1.1 pairwise/pointwise INVASE model with/without margin loss

Each row in the Table 7.1 represents the result of one model, from top to down they are:

- First row: pairwise INVASE model with margin loss, trained with margin value 0.5
- Second row: pairwise INVASE model without margin loss
- Third row: pointwise INVASE model with margin loss, trained with margin value 0.5
- Fourth row: pointwise INVASE model without margin loss

lambda	Average number of selected features	JS within one query	JS from different queries	P@10	B@10
0.04	16.7366	0.8098	0.5643	0.4445	0.4278
0.02	17	0.7014	0.7	0.4368	0.4211
0.095	17.978	0.8148	0.7641	0.4091	0.391
0.03	17	0.6615	0.6545	0.41	0.4097

Table 7.1: Result of proposed model and other three baselines

The lambda value is used to control the number of selected features per instance, but it is not able to make the number fixed. To make the performances of these models comparable, for each model we tuned the lambda value, to make the average number of selected features of different models similar.
It is not guaranteed that the selector network generates the same selected features for each document in one query, so firstly we check the JS(Jaccard Similarity) within one query and from different queries, and the NDCG score

of predictor network and baseline network by using the selected features.
From Table 7.1 we can see the introduction of the margin loss does affect the model performance. The effect of the margin loss can be seen from the value of JS. Compared with the second row in Table 7.1, the gap between JS within one query and JS from different queries in first row is much bigger. The gap in the third row is slightly bigger than in the fourth row. Compared with the pointwise INVASE model without margin loss in the fourth row, which is also the standard INVASE, the pairwise INVASE model with margin loss increases the gap difference from 0.07 to 0.2455. From the perspective of JS, we can conclude that the pairwise INVASE model with margin loss can generate a set-wise selection compared with the other three models.
JS reflects the property of the selected features, what we concern more about is the performance of the predictor network using the selected features. We can see that the performance of the predictor network is mostly better than the performance of the baseline network, which means that using a subset of features could lead to better model performance. Due to the intrinsic difference between the pointwise L2R algorithm and the pairwise L2R algorithm, we can see that the pairwise INVASE models have a better performance with respect to the NDCG score. We focus on comparing the first row and the second row in Table 7.1, the pairwise INVASE model with margin loss has the best NDCG score when the lambda value is 0.04. We can conclude that the introduction of margin loss can lead to a better performance of the predictor network by using the selection generated by the selector network.

7.1.2 performance of the set-wise selection

Our goal is to find a set-wise selection, which means that the selected features for each document in one query are the same. Since in most cases the value of JS within one query is not 1, we choose the intersection of the selected features in one query as the set-wise selection. To make the final results comparable, we set the size of the intersection to 6. The intersection is generated as follows: each document has a vote on the features, we choose features with the top 6 number of votes. We also trained an L2X model that is used as a baseline. The first four rows in Table 7.2 correspond the rows in Table 7.1. From the result in Table 7.2, we can find that the pairwise INVASE model with margin loss has the best performance.

model name	size of set-wise selection	P@10
pairwise INVASE model with margin loss	6	0.429
pairwise INVASE model without margin loss	6	0.4243
pointwise INVASE model with margin loss	6	0.3759
pointwise INVASE model without margin loss	6	0.3403
L2X model	6	0.324

Table 7.2: performance of the set-wise selection

7.1.3 pairwise INVASE model with margin loss applied on a bigger dataset

In MSLRWEB-10K each document has 136 features. The number of negative samples used in one instance in training is set to 5. We trained the pairwise INVASE model with margin loss with respect to different lambda values, the margin value was set to 0.5. From Table 7.3 we can see that the gaps between JS within one query and JS from different queries are bigger than the gaps in Table 7.1 and the JS within one query is almost 1, which means that the selected features for each document in one query are almost same, close to the set-wise selection.

pairwise INVASE model with margin loss

lambda	Average number of selected features	JS within one query	JS from different queries	P@10	B@10
0.1	64.6047	0.9736	0.4882	0.3129	0.3058
0.4	55.485	0.9512	0.4809	0.3123	0.342
0.8	21.427	0.8903	0.4157	0.3289	0.2899

Table 7.3: margin=0.5

We set the size of the set-wise selection to 6, and compare the performance of the set-wise selection of the trained models from Table 7.3 and the L2X model, and find that the pairwise INVASE model with lambda value 0.8 has the best performance.

performance of the set-wise selection

model name	size of set-wise selection	P@10
pairwise INVASE with lambda 0.1	6	0.2876
pairwise INVASE with lambda 0.4	6	0.3065
pairwise INVASE with lambda 0.8	6	0.3101
L2X model	6	0.2971

Table 7.4: performance of the set-wise selection

7.2 The effect of the number of the negative documents

Pairwise INVASE models with margin loss are trained with respect to different number of negative documents, k=10, k=15, k=20, compared with the result of using k=5 in the first row of Table 7.1.
From the following tables, we find that more negative samples used in the margin loss could lead to a more stable gap between JS within one query and JS from different queries. There is no apparent performance improvement compared with the result in the first row of Table 7.1 since the negative samples are used to force the model to generate different set-wise selections for different queries. And we can find that the number of selected features is not directly controllable through the lambda value.

lambda	Average number of selected features	JS within one query	JS from different queries	P@10	B@10
0.04	21.4076	0.6658	0.4936	0.4458	0.4204
0.045	16.6839	0.6718	0.4934	0.4306	0.4331
0.05	11.2949	0.6701	0.4807	0.4116	0.4152

Table 7.5: k=10

lambda	Average number of selected features	JS within one query	JS from different queries	P@10	B@10
0.04	18.1779	0.664	0.4991	0.4122	0.4217
0.045	19.1238	0.656	0.4913	0.4259	0.4214
0.05	18.0267	0.6743	0.5039	0.4354	0.4368

Table 7.6: k=15

lambda	Average number of selected features	JS within one query	JS from different queries	P@10	B@10
0.04	22.1683	0.6663	0.4847	0.4453	0.434
0.045	15.8033	0.6636	0.4906	0.4296	0.4452
0.05	15.6889	0.6646	0.4846	0.4325	0.4423

Table 7.7: k=20

Chapter 8

Conclusion

Summary

In this thesis, the proposed the pairwise INVASE model with margin loss can generate the set-wise interpretation for a trained LTR model. By comparing with the other baseline models the proposed model has a better performance. From the perspective of the training process, pairwise INVASE models outperform the pointwise INVASE model. And from the perspective of JS, the introduction of margin loss increases the gap between JS within one query and JS from different queries, a bigger gap means that the generated set-wise interpretation is more consistent.

Open Questions and Future Work

The average number of selected features is very sensitive to the lambda value, it is hard to control it. From Table 7.2 we can find out that the performance of the pairwise INVASE model with margin loss is slightly better than without using margin loss. To control the number of the selected features, we could sample a concrete distribution based on the output of the selector network to get a mask, instead of sampling directly. In this way, we can not only select any number of features but also train the model end-to-end. The documents are organized in one query because of the relationship to the query. The selector network focuses on generating the output of documents, without considering the query information directly. It could be helpful to integrate a learned representation of query in the training of pairwise INVASE model with margin loss.

Chapter 9

Appendix

9.1 Compared with 7.1.2: direct intersection as set-wise selection

model name	P@10
pairwise INVASE model with margin loss	0.4363
pairwise INVASE model without margin loss	0.417
pointwise INVASE model with margin loss	0.3692
pointwise INVASE model without margin loss	0.3487
L2X model	0.324

Table 9.1: performance of the set-wise selection

Bibliography

[1] A. Abid, M. F. Balin, and J. Y. Zou. Concrete autoencoders for differentiable feature selection and reconstruction. *CoRR*, abs/1901.09346, 2019.

[2] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. Learning to rank using gradient descent. In L. D. Raedt and S. Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 89–96. ACM, 2005.

[3] E. Candes, Y. Fan, L. Janson, and J. Lv. Panning for gold: Model-x knockoffs for high-dimensional controlled variable selection, 2017.

[4] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation, 2018.

[5] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[6] Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[7] B. Kang, J. Lijffijt, and T. D. Bie. Explaine: An approach for explaining network embedding-based link predictions. *CoRR*, abs/1904.12694, 2019.

[8] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[9] S. Lundberg and S. Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[10] C. Molnar. *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[11] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[12] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[13] J. Singh and A. Anand. Model agnostic interpretability of rankers via intent modelling. In M. Hildebrandt, C. Castillo, E. Celis, S. Ruggieri, L. Taylor, and G. Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 618–628. ACM, 2020.

[14] J. Singh, Z. Wang, M. Khosla, and A. Anand. Valid explanations for learning to rank models. 2020.

[15] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*, abs/1903.03894, 2019.

[16] J. Yoon, J. Jordon, and M. V. D. Schaar. Invase: Instance-wise variable selection using neural networks. In *ICLR*, 2019.