A Linear Transform Scheme for Combining Weights into Scores

S. Y. Sung

Dept. of Computer Science

Rice University

Houston, TX 77005-1892

email: *ssung@cs.rice.edu*

Abstract

Ranking has been widely used in many applications. A ranking scheme usually employs a "scoring rule" that assigns a final numerical value to each and every object to be ranked. A scoring rule normally involves the use of one to many scores, and it gives more weight to the scores that is more important. In this paper, we give a scheme that can combine weights into scores in a natural way. We compare our scheme to the formula given by Fagin. We give some additional desirable properties that weighted "scoring rules" are desirable to possess. Some interesting issues on weighted scoring rule are also discussed.

1 Introduction

Ranking has been used in many applications. Examples include multimedia databases, information retrievals, image and pictorial databases, and many others. In these applications, ranking is a pre-requisite because it is very common that users will issue *similarity searches*. Similarity is defined in terms of a similarity distance function. The smaller the similarity distance value, the more similar are two objects.

Typical similarity search queries are the similarity range query and the *k*-nearest neighbor query. A similarity range query is specified by a query object and a similarity distance range [0, ε]. A *k*-nearest neighbor query is specified by a query object and a number *k,* the *k* most similar objects to the query object are to be retrieved. There is a large literature in the area of efficient support of similarity search. For example, two of recent algorithms on k-nearest neighbor search can be found in [SK98] and [Fa98]. In all of similarity searches, ranking or "scoring rule" is used for the purpose of sorting qualified objects, or filtering out unqualified objects. In [FW97], Fagin and Wimmers have addressed the problem of "incorporating weights into scoring rules", because it is often desirable and/or necessary that extra weight can be assigned to more important attributes (or subqueries, judges, etc) in order to get a better or fair result. This problem is also our main focus in this paper.

Fagin and Wimmers [FW97] give a simple but elegant formula for incorporating weights into ranking rules. Their formula has several desirable properties (details will be described in Section 3). Most important, their formula can be applied to almost any underlying ranking rules straightforwardly. However, the formula also causes some undesirable problems, some of them are noted by the authors themselves. We will see details in Section 3 too. To study and overcome the problems encountered by Fagin's method is one of our motivations.

Our main contributions in this paper are (1) study Fagin's method and the reason behind its problems, (2) propose our solution for the problem of "incorporating weights into scores", and (3) discuss some other interesting issues related to this topic.

The rest of paper is organized as follows. The problem formulation is given in Section 2. In Section 3 we present a brief introduction to the solution of incorporating weights into scores proposed by Fagin and Wimmers. We listed some of the drawbacks of their method. In Section 4, we propose a new method. Our method is based on linear transform, a natural way of extending the ranking rule from the unweighted case to weighted case. We show that our method has overcome the drawbacks of the previous work. In particular, we apply our method on the *minimum* ranking rule, and show the advantages of our method over two other methods. In Section 5, we discuss some interesting related problems. Our conclusions are given in Section 6.

2 Problem Formulate

The general ranking rule is as follows. Each object *X* has a number of scores associated with it. A "ranking rule" is applied to *X* to generate an overall score of *X*. The overall score is then used to rank *X*. A ranking rule is normally a mathematical function which uses an object's scores as arguments.

Let the number of scores associated with an object X be denoted as $(x_1, x_2, \ldots, x_n)$. Without loss of generality, we can assume that

(1) $0 \le x_i \le 1$, for all $i=1,\ldots,n$, where 1 represents a perfect match.
(2) the higher the overall score is, the higher the rank has.

A popular ranking rule is to use geometric distance function of the following:

$$f(x_1, x_2, \ldots, x_n) = (x_1^\alpha + x_2^\alpha + \ldots + x_n^\alpha)^{\frac{1}{\alpha}}, \quad \text{where } \alpha = 1, 2, \ldots, \infty \qquad (2.1)$$

In particular, we have

(i) the *sum* rule: when $\alpha = 1$, $f(x_1, x_2, ..., x_n) = (x_1 + x_2 + ... + x_n)$

(ii) the Euclidean distance: when $\alpha = 2$, $f(x_1, x_2, ..., x_n) = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$

Other examples of ranking rules often used in multimedia database retrieval and information retrieval are: *minimum* and *maximum*.

(iii) the *minimum* rule: $f(x_1, x_2, ..., x_n) = \min(x_1, x_2, ..., x_n)$ (2.2)

(iv) the *maximum* rule: $f(x_1, x_2, ..., x_n) = \max(x_1, x_2, ..., x_n)$ (2.3)

These two rules are attractive because they are equivalent to the conjunctive and disjunctive in the standard fuzzy semantics [BG73, Za65, Zi96].

Other forms of ranking rules can be found in many other areas, such as economics [AR86] and sports competition [BP94].

In practice, it is obvious that in many cases different weights need to be assigned to different attributes, different judges, etc. For example, in multimedia database, the user might like to give more weight to color than shape in a search for similar pictures. Therefore, It is natural to ask for some meaningful and systematic way to combine weights and ranking rules. Fagin and Wimmers [FW97] have proposed a formula for incorporating weights into ranking rules. Their formula has many desirable characteristics (details to be described in the Section 3 shortly). However, the formula also has some undesirable features, and that inspires us to study the reasons and to find other alternate solutions.

3 Fagin and Wimmers's solution

We now describe the solution of Fagin and Wimmers, with more details can be found in [FW97].

Given an unweighted rule $f(x_1, x_2, ..., x_n)$, and a set of weights $\theta = \{\theta_1, \theta_2, ..., \theta_n\}$, where each θ_i is the weight respect to the score x_i. Assume that $\theta_1, \theta_2, ..., \theta_n$ are all nonnegative and sum to one, and without loss of generality, $\theta_1 \geq \theta_2 \geq ... \geq \theta_n$. A corresponding weighted rule is generated by the following formula:

$$\begin{aligned} f_{(\theta_1, \theta_2, ..., \theta_n)}(x_1, x_2, ..., x_n) = &(\theta_1 - \theta_2) f(x_1) + \\ &2(\theta_2 - \theta_3) f(x_1, x_2) + \\ &3(\theta_3 - \theta_4) f(x_1, x_2, x_3) + \\ &... + \\ &n\theta_n f(x_1, ..., x_n) \end{aligned} \tag{3.1}$$

The formula in above is referred to as "FW formula" in this paper.

An intuitive explanation of the FW formula is that x_1 has an extra weight of $(\theta_1-\theta_2)$ than all the other x_i, so we first compute this excessive score for x_1. This gives the first term $(\theta_1-\theta_2)f(x_1)$ in the formula. Similarly, both x_1 and x_2 have an extra weight of $(\theta_2-\theta_3)$ than the other x_is, so we then compute this excessive score for x_1 and x_2, and get the second term in the formula. Continue in this way, we finally get the last term of the formula.

The FW formula is simple in format, and elegant in intuition. It has the beauty that it works with any underlying unweighted rule in a straightforward manner.

The weighted rule f_θ generated from FW formula has the following desiderata:

D1: $f_\theta(x_1,...,x_n) = f(x_1,...,x_n),\ if\ \theta_1 = \theta_2 = ... = \theta_n$. That is, if all weights in θ are equal, then the "weighted" rule f_θ coincides with the "unweighted" rule f.

D2: $f_{(\theta_1,...,\theta_{n-1},0)}(x_1,...,x_n) = f_{(\theta_1,...,\theta_{n-1})}(x_1,...,x_{n-1})$. That is, if a particular argument has zero weight, then that argument can be dropped without affecting the value of the result.

D3: $f_{(\theta_1,...,\theta_n)}(x_1,...,x_n)$ is a continuous function of $\theta_1,\theta_2,...,\theta_n$.

The weighted rule f_θ obtained from FW formula inherits many properties of the underlying unweighted rule f. For example, one important such property is *monotonic*. We write $X \geq X'$ for two objects X and X' if $x_i \geq x_i'$ for each i $(i=1,\ldots,n)$, and $X > X'$ if $x_i > x_i'$ for each i, $(i=1,\ldots,n)$. We say an unweighted rule f is *monotonic* if $f(X) \geq f(X')$ whenever $X \geq X'$. Also, f is *strictly monotonic* if $f(X) > f(X')$ whenever $X > X'$. Analogously, a weighted rule is monotonic (strictly monotonic) if each f_θ is monotonic (strictly monotonic). The property of monotonic means if each individual score of X is at least as big as the corresponding score of X', then the overall score of X should be as least as big as the overall score of X'. A rule is monotonic but not strictly monotonic means there is a portion of the domain where the ranking rule is insensitive; and such a rule might be considered as undesirable. In next section, we will mention a few more properties.

Unfortunately, there are some serious drawbacks with FW formula.

- Equivalence property is not preserved.

We call two ranking rules f and g *equivalent* if $f(x)<f(y)$ whenever $g(x)<g(y)$ and vice versa. For example, The function *sum:* $f(x_1,...,x_n) = x_1 + ... + x_n$ and function *average:* $f(x_1,...,x_n) = \dfrac{x_1 + ... + x_n}{n}$ are equivalent. The functions $f(x_1,...,x_n) = \sqrt{x_1^2 + ... + x_n^2}$ and $f(x_1,...,x_n) = x_1^2 + ... + x_n^2$ are also equivalent.

In ranking, it is the relative, rather than the absolute, score that matters. For example, to find the winner of two arbitrary objects, using the function *sum* or *average* makes no difference -- the winner is always the same. Thus, the equivalence property is very useful in ranking since it means two equivalent rules will give the same results as far as ranking is concerned. In practice, which rule to use (among several equivalent rules) is normally a consideration of efficiency, or simply a matter of convenience.

Under FW formula, equivalent unweighted-rules are often not generating equivalent weighted-rules. For example, assume $\theta_1 \geq \theta_2$ the *weighted sum* is $f_{(\theta_1,\theta_2)}(x_1, x_2) = (\theta_1 - \theta_2)x_1 + 2\theta_2(x_1 + x_2) = x_1 + 2\theta_2 x_2$ and *weighted average* is $f_{(\theta_1,\theta_2)}(x_1, x_2) = (\theta_1 - \theta_2)x_1 + 2\theta_2 \frac{x_1 + x_2}{2} = \theta_1 x_1 + \theta_2 x_2$ (for the sake of simplicity, we only use two scores). Let $\theta_1=2/3$ and $\theta_2=1/3$, we get *rank(0.4,0.8) > rank(0.8,0.1)* for weighted sum while *rank(0.4,0.8)< rank(0.8,0.1)* for weighted average.

- Lack of natural generalization and variation.

For a given unweighted rule, FW formula generates only one corresponding weighted rule. Also, this generalization is not a natural extension of the underlying unweighted rule. For example, the FW formula generates the following weighted rule for Euclidean distance function (again, we assume only 2-score case, and $\theta_1 \geq \theta_2$):

$$f_\theta(x) = (\theta_1 - \theta_2)x_1 + 2\theta_2\sqrt{x_1^2 + x_2^2}$$

The above rule is not a natural generalization of the Euclidean distance. It also does not give the rich variation that a weighted Euclidean distance can be. For example, both

$f_\theta(x) = \sqrt{\theta_1 x_1^2 + \theta_2 x_2^2}$ and $f_\theta(x) = \sqrt{\theta_1^2 x_1^2 + \theta_2^2 x_2^2}$ are two different natural extension of the unweighted Euclidean distance, but none of them is equivalent to the above function.

- Aggregation anomaly

We show an illustrative example here, and give more details of this part in Section 5.

	w_1	w_2
θ_1	0.8	0.6
θ_2	0.6	0.8

Object ***X***

	w_1	w_2
θ_1	0.6	0.8
θ_2	0.8	0.6

Object ***X'***

Assume $w_1=0.6$, $w_2=0.4$, $\theta_1=0.6$ and $\theta_2=0.4$. Suppose the underlying rule used is *sum*. The overall score for object X is computed as $(0.36-0.24)\times0.8+0+3\times(0.24-0.16)\times2.0+4\times0.16\times2.8=2.368$. However, if we compute by row first and then followed by column. We get $(0.6-0.4)\times0.8+2\times0.4\times1.4=1.28$ (for 1st row), $(0.6-0.4)\times0.6+2\times0.4\times1.4=1.24$ (for 2nd row), and then by column we get the overall score $(0.6-0.4)\times1.28+2\times0.4\times2.52=2.272$.
Similarly, we can get the overall scores for object X' are *2.392* and *2.264*, respectively.

This example shows two things: (1) if we apply FW formula to individual data directly, the result might be different from if we apply the formula aggregately (row by column in this example), and (2) the two weighted rules (direct versus aggregate) are not equivalent.

- Computational efficiency

Since the FW formula needs to compute the underlying function n times, it is in general less efficient than a weighted rule that computes the underlying function only once (or less than n times).

- Embedding into search method

A ranking rule is often used together with a search method, especially in database applications. For example, in finding the nearest k objects in similarity search [SK98], or the top k answers in multimedia search [Fa98], or finding the relevant documents in information retrieval [FM98], etc. Fagin has shown that the weighted rule generated according to FW formula can be straightly used on several search algorithms [Fa98, FM98] with no or little modification. However, it is important to note that there are many other search algorithms where we do not know whether we can convert an algorithm for the unweighted case into an algorithm for the weighted case, without drastically changing the structure or affecting the performance. However, further discussion on this particular issue is beyond our scope here.

We find two reasons that might explain the above problems caused by FW formula. The first one is because the weights in FW formula do not tightly embed into the underlying function. That is, the weights are not mixed with the scores directly, they are simply outside the functions. This makes the weighted rules unnatural and lack of variety.

The second reason is due to the partial set evaluation. Note that the FW formula uses only subsets in its first $(n-1)$ terms, and only the last term uses the full set of $\{x_1,x_2,\ldots,x_n\}$. This partial set evaluation is the direct cause for most of the above problems, e.g., the loss of equivalence. It also accounts for the problem of *counterfactual* [Le73], where one is forced to consider worlds that are commonly known as impossible. One such example described in [FW97] is in the diving competition where the overall score is obtained by eliminating the top and bottom scores, then summing the remaining scores and multiplying by the degree of difficulty of the dive. Suppose there are seven judges, and each judge i gives a score x_i. The unweighted rule is as follows:

$f(X) = d \cdot g(X)$

where d is the degree of difficulty of the dive, and

$$g(X) = \sum_{i=1}^{7} x_i - \max\{x_1,\ldots,x_7\} - \min\{x_1,\ldots,x_7\} \tag{3.2}$$

The weighted rule according to FW formula need not only evaluate $f(X)$ when X is over the full set $I=\{x_1,x_2,\ldots,x_7\}$, but also the evaluation of $f(X)$ when X is over a proper subset of *I*. It is strange that even if we know there are always exactly seven judges, we are forced to consider the situation where there are fewer judges. In addition, it is not clear how to properly handle the case where there are less than three judges.

4 Our approach

The essential principle of a ranking rule is that the highest ranked object should have the shortest distance from the perfect point, and the farther from the perfect point the lower the rank will have. The perfect point in our case is $A^+=(1,1,\ldots,1)$. Analogously, we can select $A^-=(0,0,\ldots,0)$ as the negative perfect point, and in general the farther an object is from the negative perfect point the higher the rank will be.

The rational for our approach is that the strength of a score is in general proportional to its weight. In addition, we intend to use full set evaluation, and directly combine weights with scores.

The idea of our method is based on *transformation*. The simplest form of our method is linear transform.

- Linear transform

Suppose the underlying unweighted rule is $f(x_1,\ldots,x_n)$. Let the weights be $\theta=(\theta_1,\ldots,\theta_n)$, where $\theta_1,\theta_2,\ldots,\theta_n$ are all nonnegative and sum to one. For an object $X=(x_1,\ldots,x_n)$, the overall score of *X* under our weighted rule is

$$f_\theta(X)=f(\theta_1 x_1,\theta_2 x_2,\ldots,\theta_n x_n) \tag{4.1}$$

For example, if $f(x_1,x_2,\ldots,x_n)=(x_1^\alpha+x_2^\alpha+\ldots+x_n^\alpha)^{\frac{1}{\alpha}}$, where $\alpha=1,2,\ldots,\infty$, then linear transform method will generate the corresponding weighted rule

$f_\theta(x_1, x_2, ..., x_n) = (\theta_1^\alpha x_1^\alpha + \theta_2^\alpha x_2^\alpha + ... + \theta_n^\alpha x_n^\alpha)^{\frac{1}{\alpha}}$, where $\alpha = 1, 2, ..., \infty$.

Our method preserves the equivalence property.

Theorem 4.1: If *f* and *g* are equivalent in the unweighted case, then they are also equivalent in the weighted case.

Proof: Let $f_\theta(X) < f_\theta(Y)$. Since *f* and *g* are equivalent, from (4.1), we have also $g_\theta(X) < g_\theta(Y)$.

It is obvious that our method also preserves the *monotonic* and *strict monotonic* property.

Since in ranking, it is the relative overall score that counts, we can slightly modify the first desirable property D1 as follows:

D1': If all weights in θ are equal, then the "weighted" rule f_θ is equivalent to the "unweighted" rule *f*.

The above change does not compromise the significance of the property.

Since our method is more tightly coupled with the underlying function, when we check whether a desirable property is satisfied by the weighted rule generated using our method, it is natural that we need to examine the underlying function themselves.

The property D1' is satisfied if the underlying function *f(X)* is equivalent to the function *g(X)=f(cX)* where *c* is a constant. In general we expect the function *g(X)* to be equivalent to *f(X)* if *f* is a reasonable ranking function. Because intuitively, *f(X)<f(Y)* means *X* is closer to the negative perfect point $A^-=(0,0,\ldots,0)$ than *Y*, so we expect that *cX* is also closer to A^- than *cY* for any constant *c* (technically we can restrict *c* to be <1).

For all the functions we mentioned so far, namely (2.1)~(2.3) & (3.2), they all have the character of *f(cX) = cf(X)* when *c* is some constant. Thus, they all satisfy property D1'.

Property D2, in our case, means the underlying function f has the character that a zero score will contribute nothing to the overall score. All functions we mentioned, except the *min*, has this character.

Property D3 is true for any underlying function *f* that is continuous.

The overall score obtained by (4.1) is normally in the range [0, ∞). It is sometimes desirable to have the overall score within a range [0,1] with maximum and minimum achieved by A^+ and A^-, respectively.

Definition. A ranking rule f is in *standard format* if $0 \le f(X) \le 1$, $f(A^-)=0$ and $f(A^+)=1$.

The following lemma shows a weighted rule obtained by our method can always be converted into a standard format.

Lemma 4.2. Let $g_\theta(X) = \dfrac{f_\theta(X) - f_\theta(A^-)}{f_\theta(A^+) - f_\theta(A^-)}$. Then g_θ is equivalent to f_θ, and g_θ is in standard format.

- Other transforms

Besides linear transform, other type of transformations might be useful and meaningful. For example, in Euclidean distance, a linear transform $x_i \to \theta_i x_i$ gives the weighted rule $\sqrt{\sum_{i=1}^n \theta_i^2 x_i^2}$. A square-root transform $x_i \to \sqrt{\theta_i} x_i$ would give another weighted rule $\sqrt{\sum_{i=1}^n \theta_i x_i^2}$. Both have been used in practice [BM+97, CH+95, NB+93, SFW83], both seem to be natural generalization of the unweighted Euclidean rule, but they are not equivalent.

Example of other type of transform includes exponential transformation $x_i \rightarrow x_i^{\theta_i}$. If the underlying function is a form of product, e.g. $f(X) = x_1 \cdot x_2 \cdots x_n$, then exponential transformation is more meaningful to use.

At the first glance, it seems that a weighted *minimum* rule can not be generalized using our method. But it is interesting to know that our method indeed generates a very natural weighted *min* rule.

- Weighted min rule

A direct application of the linear transform to min function gives the weighted rule $\min\{\theta_1 x_1,...,\theta_n x_n\}$ which is obvious not valid. The reason is the concept of *minimum* is in direct conflicting with the concept of *weighting*.

As explained at the beginning of this Section, we can take the *minimum* from the point of view of either A^- or A^+. It is true that

$$\min\{x_1,...,x_n\} = 1 - \max\{1 - x_1, ..., 1 - x_n\} \tag{4.2}$$

This shows that if we view minimum from the point of A^+, we can use max instead of min. Since the concept of *max* has no conflict with *weighting*, we can apply our method to it and expect to generate a reasonable weighted rule.

Applying the linear transform, the point A^+ is transformed to $(\theta_1, \ldots, \theta_n)$. Thus, under our linear transform method, the weighted min rule is formulated as

$$f_\theta(X) = 1 - \max\{\theta_1 - \theta_1 x_1,...,\theta_n - \theta_n x_n\} \tag{4.3}$$

Using lemma 4.2, we convert (4.3) into a standard format of the weighted min rule as following:

$$f_\theta(X) = 1 - \frac{1}{M}\max\{\theta_1 - \theta_1 x_1, \ldots, \theta_n - \theta_n x_n\} \tag{4.4}$$

where $M = \max\{\theta_1, \ldots, \theta_n\}$.

- Comparison of weighted min rules

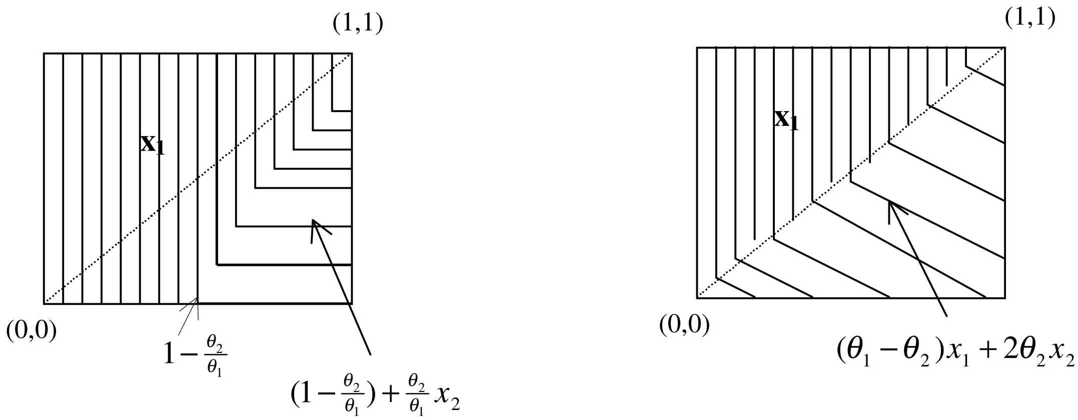
We now compare our weighted min rule with the one obtained by FW formula. It is instructive to also include some other weighted min function in the comparison, like the one given by Dubois and Prade [DP86] which has been used in fuzzy set theory:

$$f_\theta(X) = \min_{i \in \{1,\ldots,n\}}\{\max\{1 - (\frac{\theta_i}{M}), x_i\}\} \tag{4.5}$$

where $M = \max\{\theta_1, \ldots, \theta_n\}$.

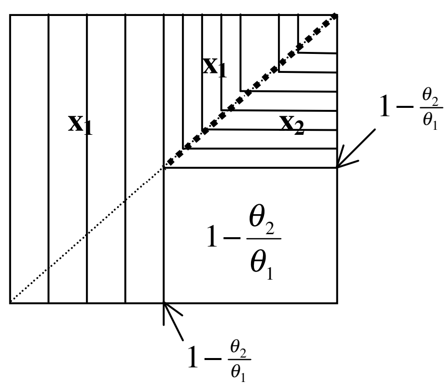
All three rules satisfy properties D1~D3, and monotonic. But (4.5) is not strict monotonic while the other two are.

The following figure is a 2D visualization of the three rules. In the Figure, assume the weights are $\theta_1 = 2/3$ and $\theta_2 = 1/3$.



(a) Linear transform

(b) FW formula



(c) Formula (4.5)

Note that:

- All three rules has $min = x_1$ when $x_1 \le x_2$.
- Formula (4.5) gives $min = 1-\frac{\theta_2}{\theta_1}$ when $x_1 \ge 1-\frac{\theta_2}{\theta_1} \ge x_2$. This is rather undesirable.
- When $x_2 > x_1$, under FW, both x_1 and x_2 contribute to the overall score.
- Under linear transform, there is exactly one, either x_1 or x_2, contributing to the overall score.

Figure 4.1 2D contours of three *weighted min* ranking rules

5 Other related problems

So far, the weights we discussed are from 1-dimension only. In some applications, the weights can be formed in 2-dimension or higher. For example, in a search for similar events, one set of weights is given for the attributes of the event, another set of weights is given for the time of that event occurred (the more recent, the higher weight). A 2-dimensional weights assignment can be viewed as below:

X	w_1	w_2	…	w_m
θ_1	x_{11}	x_{12}	…	x_{1m}
θ_2	x_{21}	x_{22}	…	x_{2m}
:	…	…	…	…
θ_n	x_{n1}	x_{n2}	…	x_{nm}

(a)

X	w_1	w_2	…	w_m	
θ_1	x_{11}	x_{12}	…	x_{1m}	y_1
θ_2	x_{21}	x_{22}	…	x_{2m}	y_2
:	…	…	…	…	…
θ_n	x_{n1}	x_{n2}	…	x_{nm}	y_n

(b)

Table 5.1 A *2D*-weights example

5.1 Multi-dimensional weights

Given a *2D* type of weights, as shown in above table 5.1(a). We can evaluate the overall score for object *X* on *individual score* as follows:

$$f_{\theta w}(X) = f_{(\theta_1 w_1,\ldots,\theta_n w_m)}(x_{11},\ldots,x_{nm})$$

We call the above *joint* weighted rule.

Another way is to have an aggregated score first (e.g., by aggregating over row), then evaluate on the aggregated score to get the overall score. That is,

$$f_{\theta w}(X) = f_{(\theta_1,\ldots,\theta_n)}(y_1,\ldots,y_n), \quad \text{where } y_i = f_{(w_1,\ldots,w_m)}(x_{i1},\ldots,x_{im})$$

We name the above *aggregate* weighted rule.

There are three reasons that we want the joint method and the aggregate method to be equivalent:

(1) for security and/or confidential purpose, sometimes individual data is not disclosed, only summary data (i.e. aggregated data) is shown,
(2) in some cases of data warehousing [Wi95], only summary data can be obtained quickly, to access individual data can be slow and expensive, and
(3) even when both individual and summary data are available, it is usually more efficient to compute from the summary data compared to compute directly from individual data [MQM97].

The two rules in fact, under linear transform method, are the same for many cases, including (2.1), (2.3) and (4.2). We show the case of (4.2) in the following (the other two cases are straightforward, so omitted here):

For *min* rule (4.2), we have

$$y_i = f_{(w_1,\ldots,w_m)}(x_{i1},\ldots,x_{im}) = 1 - \frac{1}{M}\max\{w_1(1-x_{i1}),\ldots,w_m(1-x_{im})\} \qquad \text{(by (4.4))}$$

where $M = \max\{w_1, \ldots, \mathrm{w}_m\}$.

Thus, *aggregate* min rule is

$$f_{\theta w}(X) = f_{(\theta_1,\ldots,\theta_n)}(y_1,\ldots,y_n) = 1 - \frac{1}{M'}\max\{\theta_1(1-y_1),\ldots,\theta_n(1-y_n)\}$$

where $M' = \max\{\theta_1, \ldots, \theta_n\}$.

It is not hard to verify that the above is equal to the *joint* min rule.

However, the two rules are not equivalent under FW formula, one example of the *sum* weighted rule is shown in Section 3. The following is another example, for the case of *min* weighted rule:

X	w_1=.6	w_2=.4	y
θ_1=.6	.8	.6	.64
θ_2=.4	.6	.8	.6

X'	w_1=.6	w_2=.4	y
θ_1=.6	.7	.6	.62
θ_2=.4	.7	.7	.7

Under FW, the *aggregate min* has overall score *0.608* for *X*, and *0.62* for *X'*. The *joint min* has overall scores *0.624* and *0.612* for *X* and *X'*, respectively. This shows that the two rules, *aggregate min* and *joint min* are not equivalent under FW formula.

Another interesting ranking related problem is what we call *partial set ranking* problem.

5.2 Partial set ranking

In real life, it often happens that a ranking is required to be completed within a short time period, but the number of evaluators available is much smaller than the number of objects that need to be evaluated. Examples are proposals submitted for a project, papers submitted to a conference, etc. Under the circumstance, it is usually very difficult, or even impossible, to ask every evaluator to evaluate every proposal. As a result, each object (proposal etc.) is normally evaluated by a small number of evaluators. In practice, we also like to assign weights to evaluators according to some measurement such as their experience or professional knowledge.

Without further elaboration, we can illustrate the problem caused by the partial set ranking by the following example in Table 5.2.

	E1(*5*)	E2(*5*)	E3(*4*)	E4(*1*)	***Average***	***Weighted average***
X	*10*	*6*			*8*	*8*
Y	*10*		*6*		*8*	*8.2*
Z		*9*		*5*	*7*	*8.3*

Table 5.2 An example of partial set ranking

In example given in table 5.2, there are four evaluators E1~E4, with weights *5, 5, 4* and *1*, respectively. Intuitively, object *X* and object *Y* should be ranked equally, and they should be ranked higher than object *Z*, as reflected by their average overall scores. However, the weighted average tells a different story that is contradicting to our intuition. We call this problem the *partial ranking anomaly*. How to solve the partial ranking anomaly problem, or to a larger extent, how to conduct the partial ranking in a "fair" manner, remains to be seen.

6 Conclusions

There is a large body of algorithms using ranking for filtering and/or sorting purpose. However, there is a lack of systematic way for combining weights into account in the ranking process. Fagin and Wimmers developed a formula which can take assigned weights into any ranking rules in a straightforward way. However, there are certain shortcomings we discovered in their method. We proposed a new method that has a few desirable properties, for example, (1) it preserves *equivalence*, (2) it can be used for multi-dimensional weights, and (3) it is a natural extension to the underlying unweighted case.

Our method is not a mechanical formula, it is a "conceptual" method in the sense that sometimes the underlying function needs to be taken into consideration as well. A good example is the construction of a *weighted min* function using our method, which has been shown in Section 4.

We also described an interesting but important related issue, the partial set ranking problem. Its solution remains to be found in the future.

Acknowledgments

We thank Moshe Vardi for providing feedback on a draft of this paper.

References

[AR86] K. Arrow and H. Raynaud, "Social Choice and Multicriterion Decision-Making", MIT Press, Cambridge (1986).

[BG73] R. Bellman and M. Giertz, "On the Analytic Formalism of the Theory of Fuzzy Sets", *Information Sciences* **5** (1973), pp. 149-156.

[BM+97] A. Bimbo, M. Mugnaini, P. Pala, and F. Turco, "PICASSO: Visula Querying by Color Perspective Regions", Proc. of *the second Intl. Conference on Visual Information Systems*, Dec 1997, pp. 125-131.

[BP94] G. Bassett and J. Persky, "Rating Skating", *J. of the American Statistical Association* **89**, No. 427 (1994), pp. 1075-1079.

[CH+95] M. Carey, L. Haas *et al*, "Towards Heterogeneous Multimedia Information Systems: the Garlic Approach", *RIDE '95* (5th Int'l Workshop on Research Issues in Data Engineering) (1995), pp. 124-131.

[DP86] D. Dubois and H. Prade, "Weighted Minimum and Maximum Operations in Fuzzy Set Theory", *Information Sciences* **39** (1986), pp. 205-210.

[Fa98] R. Fagin, "Fuzzy Queries in Multimedia Database Systems", In Proc. *1998 ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

[FM98] R. Fagin and Y. Maarek, "Allowing Users to Weight Search Terms in Information Retrieval", *IBM Research Report* RJ 10108, March 1998.

[FW97] R. Fagin and E. Wimmers, "A Formula for Incorporating Weights into Scoring Rules", In Proc. of *1997 International Conference on Database Theory*, pp. 247-261. (also available in http://www.almaden.ibm.com/cs/people/fagin/)

[Gi73] A. Gibbard, "Manipulation of Voting Schemes: A General Result", *Econometrica* **41**, 4 (1973), pp. 587-601.

[Le73] D. Lewis, "Counterfactuals", Harvard University Press, Cambridge, Mass. (1973).

[MQM97] I. Mumick, D. Quass and B. Mumick, "Maintenance of Data Cubes and Summary Tables in a Warehouse", Proc. of *1997 ACM SIGMOD*, pp. 100-111, Tucson, Arizona.

[NB+93] W. Niblack, R. Barber, *et al*, "The QBIC Project: Querying Images by Content using Color, Texture and Shape", *SPIE Conference on Storage and Retrieval for Image and Video Databases* (1993), pp. 173-187. QBIC Web Server is http://wwwqbic.almaden.ibm.com/

[SFW83] G. Salton, E. Fox, and H. Wu, "Extended Information Retreival", *Comm. ACM* **26**, 12 (1983), pp. 1022-1036.

[SK98] T. Seidl and H. Kriegel, "Optimal Multi-Step k-Nearest Neighbor Search", Proc. of *1998 ACM SIGMOD*, pp. 154-165, Seattle, Washington.

[Wi95] J. Widom, "Research problems in data warehousing", *CIKM*, November 1995. Invited paper.

[Za65] L. Zadeh, "Fuzzy Sets", *Information and Control* **8** (1965), pp. 338-353.

[Zi96] H. Zimmermann, Fuzzy Set Theory, Third Edition, Kluwer Academic Publishers, Boston (1996).