Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

# A Mathematical Programming-Based Approach to Determining Objective Functions from Qualitative and Subjective Comparisons

**Takayuki Yoshizumi**
IBM Research - Tokyo
yszm@jp.ibm.com

## Abstract

The solutions or states of optimization problems or simulations are evaluated by using objective functions. The weights for these objective functions usually have to be estimated from experts' evaluations, which are likely to be qualitative and somewhat subjective. Although such estimation tasks are normally regarded as quite suitable for machine learning, we propose a mathematical programming-based method for better estimation. The key idea of our method is to use an ordinal scale for measuring paired differences of the objective values as well as the paired objective values. By using an ordinal scale, experts' qualitative and subjective evaluations can be appropriately expressed with simultaneous linear inequalities, and which can be handled by a mathematical programming solver. This allows us to extract more information from experts' evaluations compared to machine-learning-based algorithms, which increases the accuracy of our estimation. We show that our method outperforms machine-learning-based algorithms in a test of finding appropriate weights for an objective function.

## Introduction

In such fields as optimization, simulation, and computer gaming, the solutions or states must often be evaluated by domain experts. One of the ways to model implicit knowledge and the intuitions of domain experts is to design an objective function (or an evaluation function). In typical cases, this objective function is represented as a weighted sum of several metrics. Objective functions should appropriately reflect the experts' knowledge and intuitions, which is usually done by adjusting the weights for each objective term. Once an appropriate objective function has been obtained, various kinds of optimization systems or simulation systems can be run without further involvement of the experts. The problem dealt with in this paper is to find good weights for an objective function from the experts' evaluations.

For experts, it is intuitive and natural to compare pairs of solutions and qualitatively evaluate them with such descriptions as "Solution $i$ is *similar to* solution $j$", "Solution $i$ is *better than* solution $j$", or "Solution $i$ is *far better than* solution $j$". Even when experts evaluate a single solution, they must have a reference solution in mind, and compare the single solution with that benchmark in their mind, perhaps thinking "This solution is *better than* the benchmark." Therefore, from the viewpoint of level of measurement (Stevens 1946), it is reasonable that experts' evaluations should be measured with an ordinal scale rather than an interval scale or ratio scale. Using an ordinal scale, an expert can compare a pair of solutions and qualitatively evaluate which is better, but cannot quantitatively evaluate the absolute degree of the differences between the two solutions. Even with a pair of evaluations such as "Solution $i$ is *better than* solution $j$" and "Solution $s$ is *far better than* solution $t$", we cannot conclude that the relative degree of solution $s$ against solution $t$ is, for example, two times larger than that of solution $i$ against solution $j$. At the same time, another expert might feel that "Solution $i$ is *far better than* solution $j$", even though the first expert only said "Solution $i$ is *better than* solution $j$". This implies that experts' evaluations can be subjective.

Some kinds of machine learning such as learning-to-rank (Liu 2009) may be suitable for this problem. However, sometimes evaluations made by an expert may implicitly include more information than standard machine-learning-based algorithms can extract. For example, we cannot conclude that the degree of "*far better than*" is two times larger than that of "*better than*", but we can say that the degree of "*far better than*" is at least larger than that of "*better than*". This kind of information cannot be used by existing machine-learning-based algorithms. Since it is usually time consuming for experts to evaluate solutions, it is desirable to extract as much information as possible from their evaluations. Also, existing machine-learning-based algorithms cannot appropriately deal with subjective evaluations.

In this paper, we propose a novel method for extracting more information from experts' evaluations in comparison to typical machine-learning-based algorithms. The key idea of our method is to use an ordinal scale for measuring paired differences of the objective values as well as the paired objective values. By using an ordinal scale, experts' qualitative and subjective evaluations can be appropriately expressed with simultaneous linear inequalities. As a result, the problem of finding appropriate weights of an objective function can be formulated as a linear programming (or mixed in-

teger programming) problem, which can be easily solved by a mathematical programming solver. Our method is not based on machine learning techniques, but based on mathematical programming. Therefore, we may say that our approach is a mathematical-programming-based approach to machine learning tasks. We show that our method outperforms machine-learning-based algorithms in a test of finding appropriate weights for an objective function.

## Problem Setting

Our goal is to determine an objective function from experts' evaluations. We assume that a solution has $K$ metrics, and is represented as $K$-dimensional vector $\boldsymbol{x} = (x_1, \ldots, x_K)$. An objective function $f_{\boldsymbol{w}}(\boldsymbol{x})$ is defined using the weighted sum of the objective terms as

$$f_{\boldsymbol{w}}(\boldsymbol{x}) \equiv \sum_{1 \leq k \leq K} w_k x_k, \tag{1}$$

where $w_k (\geq 0)$ is the objective weight for $x_k$. Determining the objective function is the same as determining its weight vector $\boldsymbol{w} = (w_1, \ldots, w_K)$.

We assume that the scale of measurement for the experts' evaluations is an ordinal scale with several degrees. This scale of measurement can be seen as a kind of rating scale or Likert scale (Likert 1932). For example, an expert compares two solutions $i$ and $j$, and gives a qualitative evaluation, which may be one of the three options:

- solution $i$ is *similar to* solution $j$,
- solution $i$ is *better than* solution $j$, or
- solution $i$ is *far better than* solution $j$.

(When solution $j$ is better than solution $i$, we can swap their indices.) While typical ordinal scales do not deal with degrees, it is natural for experts to include degrees with their evaluations.

It is often the case that one expert may feel that "Solution $i$ is *far better than* solution $j$", while another expert might only feel that "Solution $i$ is *better than* solution $j$". This means that the boundaries, such as between "*far better than*" and "*better than*", are somewhat subjective. Therefore, to determine an objective function (or weights $\boldsymbol{w}$) that matches experts' subjective intuitions, we need to accurately estimate the boundaries, even though they may vary from expert to expert.

Let $U$ be the set of the experts. For expert $u \in U$, let $R_{\approx}^{(u)}$, $R_{>}^{(u)}$ and $R_{\gg}^{(u)}$ be the evaluation results of expert $u \in U$, which are defined as:

- $R_{\approx}^{(u)} \equiv \{(i,j) |$ expert $u \in U$ evaluates that solution $i$ is *similar to* solution $j \}$.
- $R_{>}^{(u)} \equiv \{(i,j) |$ expert $u \in U$ evaluates that solution $i$ is *better than* solution $j \}$.
- $R_{\gg}^{(u)} \equiv \{(i,j) |$ expert $u \in U$ evaluates that solution $i$ is *far better than* solution $j \}$.

The problem dealt with in this paper is to estimate the weights $\boldsymbol{w}$ from experts' evaluations of $R_{\approx}^{(u)}, R_{>}^{(u)}$, and $R_{\gg}^{(u)}$ $(u \in U)$ defined on a solution set $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$.

While this problem seems to be a machine learning problem, we will formulate it as a mathematical programming problem. To prepare for this formulation, we will explain how to formulate machine learning problems as mathematical programming problems in the next section.

## Formulation of Machine Learning Problems as Mathematical Programming Problems

In some cases, machine learning problems will be finally mapped into mathematical programming problems (Bennett and Parrado-Hernández 2006). LP (Linear Programming) is one of the most fundamental classes of mathematical programming, where the decision variables are real numbers, and the objective function and constraints are written in linear form. The class of LP with integer variables is called MIP (Mixed Integer Programming). If a problem can be formulated as an LP or MIP problem, we can efficiently solve it with an LP/MIP solver such as CPLEX. In LP and MIP, the objective function and all constraints have to be written in linear form, which appears to be a strong limitation. In this section, using concrete examples, we will show that even LP and MIP still have strong capabilities to describe problems.

We consider a machine learning task that acquires the input-output relation $y = f_{\boldsymbol{w}}(\boldsymbol{x})$ from a training set of $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where each $\boldsymbol{x}^{(i)}$ is a $K$-dimensional vector represented as $\boldsymbol{x}^{(i)} = (x_1^{(i)}, \ldots, x_K^{(i)})$. The relation $f_{\boldsymbol{w}}$ is represented as:

$$f_{\boldsymbol{w}}(\boldsymbol{x}) \equiv \sum_{1 \leq k \leq K} w_k x_k, \tag{2}$$

where $\boldsymbol{w} = (w_1, \ldots, w_K)$ is a weight vector. This means that obtaining $f_{\boldsymbol{w}}$ is the same as determining $\boldsymbol{w}$. Using this task as an example, we will formulate it both from the machine learning perspective and from the mathematical programming perspective.

### Machine-learning-based approach

First, we will deal with this problem as a machine learning problem. If the loss function is to minimize the square error, then we can regard this problem as a standard machine learning problem, and a natural formulation is:

$$\min_{\boldsymbol{w}} \sum_{1 \leq i \leq N} \| f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - y^{(i)} \|^2. \tag{3}$$

An optimal $\boldsymbol{w}$ can be calculated by

$$\boldsymbol{w} = (X^t X)^{-1} X^t Y, \tag{4}$$

where $X$ and $Y$ are defined as

$$X \equiv \begin{bmatrix} x_1^{(1)} & \cdots & x_K^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \cdots & x_K^{(n)} \end{bmatrix}, \quad Y \equiv \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}. \tag{5}$$

### Mathematical Programming-based approach

Next we formulate the same problem as a mathematical programming problem. Here, a mathematical programming

problem is referring to a problem where an objective function is minimized (or maximized) under some constraints defined as equalities or inequalities. If the loss function is not to minimize the square error, but to minimize the absolute error, this problem can be written as an LP problem in the form

(MP1)

obj:

$$\min_{\boldsymbol{w}} \sum_{1 \le i \le N} z_i \tag{6}$$

sub. to:

$$-z_i \le f_{\boldsymbol{w}}(x^{(i)}) - y^{(i)} \le z_i,\ i = 1, \ldots, N. \tag{7}$$

In LP, minimizing the square error is not in principle possible, because the objective function and all constraints have to be written in linear form. However, we can approximately deal with the square error by using standard techniques for LP formulations (Dantzig 1998). For example, we can approximately incorporate the square error by approximating a quadratic function with a piecewise linear function as

(MP2)

obj:

$$\min_{\boldsymbol{w}} \sum_{1 \le i \le N} z'_i \tag{8}$$

sub. to:

$$-z_i \le f_{\boldsymbol{w}}(x^{(i)}) - y^{(i)} \le z_i,\ i = 1, \ldots, N, \tag{9}$$

$$\max_j \{a_j z_i + c_j\} \le z'_i, \qquad i = 1, \ldots, N. \tag{10}$$

Eq. (10) represents the piecewise linear function that approximates a quadratic function. Its coefficients of $a_j$ and $c_j$ should be appropriately set according to the corresponding quadratic function. Since the "max" function can be interpreted as a set of linear functions, this function can be used in an LP formulation. As shown by this example, LP can deal with a square error problem that appears to be impossible to represent in an LP formulation. In addition, various other kinds of functions, such as the $\epsilon$-insensitive error or a Huber function, can be used in LP formulations.

### Improving the generalization ability in mathematical programming

Improving the generalization ability (or avoiding overfitting) is an ever-present problem in the machine learning field. By adding a regularization term to the loss function, we can implement a variable selection capability, and the generalization ability can be improved, or overfitting can be avoided. The example of Eq. (11) uses a regularization term of the L2 norm.

$$\min_{\boldsymbol{w}} \sum_i \|f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - y^{(i)}\|^2 + \lambda \|\boldsymbol{w}\|^2. \tag{11}$$

An optimal $\boldsymbol{w}$ for this problem can be calculated by using

$$\boldsymbol{w} = (\lambda I + X^t X)^{-1} X^t Y, \tag{12}$$

where $I$ is the unit matrix, and $\lambda$ is a hyperparameter that controls the balance between the error term and the regularization term.

In contrast, in the mathematical programming field, the regularization can be implemented by using standard techniques for MIP formulations (Johnson, Nemhauser, and Savelsbergh 2000). Some mathematical programming problems that have integer variables as well as real number variables are called MIP problems. While MIP problems are more difficult to solve compared to LP problems, they can still be accurately solved by a state-of-the-art MIP solver such as CPLEX. By introducing binary integer variables $v_k (1 \le k \le K)$, we can add a regularization ability to the formulation of (MP1) as

(MP3)

obj:

$$\min_{\boldsymbol{w}} \sum_{1 \le i \le N} z_i + \lambda \sum_{1 \le k \le K} v_k \tag{13}$$

sub. to:

$$-z_i \le f_{\boldsymbol{w}}(x^{(i)}) - y^{(i)} \le z_i, \quad i = 1, \ldots, N, \tag{14}$$

$$0 \le w_k \le C v_k, \quad k = 1, \ldots, K, \tag{15}$$

$$v_k \in \{0, 1\}, \quad k = 1, \ldots, K, \tag{16}$$

$$\sum_{1 \le k \le K} v_k \le B, \tag{17}$$

where $C$ is a sufficiently large positive constant. When $w_k$ is non-zero, $v_k$ is constrained to be 1 due to Eq. (15). This means that the number of $v_k$ values that are set to 1 represents the number of used (or non-zero) $w_k$, which corresponds to the second term of the objective function (which is being minimized). In addition, we can explicitly specify the maximum number of $w_k$ to be used with a hyperparameter $B$, which represents the maximum number of $w$ to be used. This is one of the advantages of using mathematical programming.

## The Proposed Method

We propose a novel method that determines an objective function from the experts' evaluations by using a mathematical programming approach. The key idea of our method is to use an ordinal scale for measuring not only the paired objective values, but also the paired differences of the objective values. Suppose that an expert's evaluations are "Solution $i$ is *better than* solution $j$", and "Solution $s$ is *far better than* solution $t$". In this situation, the following inequalities obviously hold.

$$f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) > f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}), \tag{18}$$

$$f_{\boldsymbol{w}}(\boldsymbol{x}^{(s)}) > f_{\boldsymbol{w}}(\boldsymbol{x}^{(t)}). \tag{19}$$

Even if the above evaluations are given by the expert, we cannot conclude that the degree of "*far better than*" is, for example, two times larger than that of "*better than*". However, we can say that the degree of "*far better than*" is at least larger than that of "*better than*". Thus, we also have following inequality:

$$f_{\boldsymbol{w}}(\boldsymbol{x}^{(s)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(t)}) > f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}). \tag{20}$$

This implies that we can use the ordinal scale for measuring the paired differences of objective values as well as the paired objective values. Therefore, compared to using the ordinal scale only for measuring the paired objective values, we can extract more information from the experts' evaluations.

From this we can formulate the overall problem as a mathematical programming problem. Since Eqs. (18), (19), and (20) are linear inequalities in terms of the decision variables $\boldsymbol{w}$, we can easily incorporate it into the linear programming formulation. We introduce other decision variables that represent the boundaries of the comparative degrees such as the one between "*similar to*" and "*better than*", and the one between "*better than*" and "*far better than*". For expert $u \in U$, let $b_{u,0}$ be the boundary between "*similar to*" and "*better than*", and $b_{u,1}$ be the boundary between "*better than*" and "*far better than*". With these notations, we can describe the relationships among experts' evaluations that are expressed in Eqs. (18), (19), and (20) with these simultaneous linear inequalities:

$$\begin{aligned} -b_{u,0} < f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}) + \sigma_{ij}^{(u)} \quad &< b_{u,0}, \\ \forall (i,j) \in R_{\approx}^{(u)}, &\forall u \in U, \\ b_{u,0} < f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}) + \sigma_{ij}^{(u)} \quad &< b_{u,1}, \\ \forall (i,j) \in R_{>}^{(u)}, &\forall u \in U, \\ b_{u,1} < f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}) + \sigma_{ij}^{(u)}, & \\ \forall (i,j) \in R_{\gg}^{(u)}, &\forall u \in U, \end{aligned} \tag{21}$$

where $\sigma_{ij}^{(u)}$ is error term. Note that $b_{u,0}$ and $b_{u,1}$ are independently defined for each expert, which means that we can appropriately formulate subjective boundaries. To normalize $\boldsymbol{w}$, the following constraint is added to our formulation.

$$\sum_{1 \le k \le K} w_k = 1. \tag{22}$$

Under these inequalities (or constraints), we can obtain optimal $\boldsymbol{w}$ by minimizing this function:

$$\min_{\boldsymbol{w}} \sum_{u \in U} \sum_{(i,j) \in R^{(u)}} |\sigma_{ij}^{(u)}|, \tag{23}$$

where $R^{(u)}$ represents the union of $R_{\approx}^{(u)}, R_{>}^{(u)}$ and $R_{\gg}^{(u)}$. Minimizing the absolute value can be represented in LP problems such as (MP1). This is an LP problem, because the objective function and all of the constraints are written in linear forms. This means we can easily obtain the optimized $\boldsymbol{w}$ (and boundaries such as $b_{u,0}$ etc.) with an LP solver. (Note that the number of degrees does not matter in our formulation. Introducing additional degrees, such as $R_{\ggg}^{(u)}$ and $R_{\gggg}^{(u)}$, is straightforward.)

## Handling basis function selection

In most cases, an objective function consists of several terms, each of which corresponds to a certain metric for the application. In general, however, an appropriate basis function should be used for each objective term. For example, instead of using $x_k$ as it stands for the objective term $k$, $x_k^2$ may be more suitable if the relative contribution of the term should increase as the value increases. Alternatively, $\sqrt{x_k}$ may be better if the relative contribution of the term should decrease as the value increases.

We can incorporate the basis function selection capability into our mathematical programming formulation. We assume that the candidate basis functions for each objective term are given as input. Let $M_k$ be the number of candidate basis functions for the objective term $k$. Let $\phi_{kl}$ be the $l$-th basis function candidate for the objective term $k$. To handle basis function selection, we need to expand the definition of $\boldsymbol{w} = (w_1, \ldots, w_K)$ to $\boldsymbol{w} = (w_{11}, w_{12}, \ldots, w_{KM_K})$, where $w_{kl}$ is the weight for the $l$-th basis function of the objective term $k$. Using these notations, the revised definition of $f_{\boldsymbol{w}}$ can be written as:

$$f_{\boldsymbol{w}}(\boldsymbol{x}) \equiv \sum_{1 \le k \le K} \sum_{1 \le l \le M_k} w_{kl} \phi_{kl}(x_k) \tag{24}$$

As described in the previous section, in mathematical programming we can deal with overfitting by introducing integer variables. More generally, in mathematical programming, integer variables can allow for variable selection, which implies that we can take advantage of the integer variables to implement basis function selection.

Let $y_{kl}$ be the binary integer variable that represents whether or not the $l$-th candidate basis function for the objective term $k$ is being used. The MIP formulation for determining the optimal basis functions and weights becomes:

(MP4)

obj:

$$\min_{\boldsymbol{w}} \sum_{u \in U} \sum_{(i,j) \in R^{(u)}} |\sigma_{ij}^{(u)}| + \lambda \sum_{1 \le k \le K} \sum_{1 \le l \le M_k} v_{kl}$$

sub.to.

$$\begin{aligned} -b_{u,0} < f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}) + \sigma_{ij}^{(u)} \quad &< b_{u,0}, \\ \forall (i,j) \in R_{\approx}^{(u)}, &\forall u \in U \\ b_{u,0} < f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}) + \sigma_{ij}^{(u)} \quad &< b_{u,1}, \\ \forall (i,j) \in R_{>}^{(u)}, &\forall u \in U \\ b_{u,1} < f_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}) - f_{\boldsymbol{w}}(\boldsymbol{x}^{(j)}) + \sigma_{ij}^{(u)}, & \\ \forall (i,j) \in R_{\gg}^{(u)}, &\forall u \in U \end{aligned}$$

$$\begin{aligned} &\sum_{1 \le k \le K} \sum_{1 \le l \le M_k} w_{kl} = 1, \\ &0 \le w_{kl} \le v_{kl}, \quad k = 1, \ldots, K, l = 1, \ldots, M_k, \\ &v_{kl} \in \{0, 1\}, \quad k = 1, \ldots, K, l = 1, \ldots, M_k, \\ &\sum_{1 \le l \le M_k} v_{kl} \le B_k, \quad k = 1, \ldots, K, \end{aligned}$$

where $B_k$ represents the maximum number of basis functions for the objective term $k$. If we want to explicitly specify the maximum number of used basis function for the objective term $k$, it can be done just by setting $B_k$ to that maximum number.

## Related Work

Our method can be seen as a mathematical programming-based approach to a machine learning task. At the same time, the problem we are dealing with relates to how to handle humans' subjective evaluations, allowing us to extract objective knowledge. From this point of view, we discuss the relationship with previous work.

### Analytic hierarchy process (AHP)

AHP is one of the methodologies for decision making, and has a long research history and various kinds of applications (Forman and Gass 2001). When using AHP, pairwise comparisons are used to make decisions. In such comparisons, a rating scale method such as a Likert scale (Likert 1932) is often used, and a human decision maker will choose among several options (such as "Strongly agree", "Agree", "Neither agree nor disagree" etc.). Our method also uses pairwise comparisons and multiple degrees or options. The long history of AHP shows that it is reasonable for human experts to use such several degrees.

### Learning to rank

Learning-to-rank algorithms (Joachims 2002; Liu 2009) are for ranking items based on a partially ordered list, and one of the most typical applications of those algorithms is for adjusting a ranking result for Web search engines. Depending on the input representation, there are several approaches, and one of them is a pairwise approach. However even the pairwise approach of the learning-to-rank algorithms cannot deal with multiple degrees of differences, which is an essential difference between our method and learning-to-rank algorithms.

### SVM-based algorithms

If it is reasonable to map "*better than*" to, for example, 2–4, "*far better than*" to 4–6 and so on, some kinds of SVM-based algorithms, such as an SVM that uses the $\epsilon$-insensitive error, can be applied to the problem (Cao et al. 2006; Steinwart and Christmann 2009). However, it is difficult to determine in advance a quantitative value range for each degree. In addition, a degree of "*better than*" could be different from expert to expert. This means it is difficult to apply SVM-based algorithms to this problem. In comparison, our method can determine a degree for each expert independently, which is an essential advantage over SVM-based algorithms.

## Experiments

We conducted experiments to demonstrate that our method can precisely estimate the weights of an objective function from the evaluation results of multiple experts who have different boundaries between their degrees.

We assumed that there are three experts (Expert1, Expert2, and Expert3), and the number of degrees is five. The correct boundaries of the degrees are set as $(b^*_{1,0}, \ldots, b^*_{1,3}) = (2, 4, 6, 8)$, $(b^*_{2,0}, \ldots, b^*_{2,3}) = (1, 2, 4, 8)$, and $(b^*_{3,0}, \ldots, b^*_{3,3}) = (4, 6, 7, 8)$. For the objective function to be estimated, we used 10 as the number of objective terms, and used the same correct weight of $0.1$ for all the terms. To generate sample solutions, each term of a sample solution ($\boldsymbol{x} = (x_1, \ldots, x_K)$) was generated according to the normal distribution of $\mathcal{N}(10, 8^2)$. To generate an experts' evaluation result (such as an element of $R^{(u)}_{\approx}$, $R^{(u)}_{>}$ etc.), we picked two sample solutions (as a sample solution pair), and added some amount of random noise from $\mathcal{N}(0, 0.1^2)$ to the calculated difference of the objective function values, and categorized the result in one of $R^{(u)}_{\approx}$, $R^{(u)}_{>}$, $R^{(u)}_{\gg}$, $R^{(u)}_{\ggg}$, or $R^{(u)}_{\gggg}$ according to the boundaries of the corresponding expert.

To compare the proposed method with other algorithms, we designed two baseline algorithms:

**Baseline 1.** For the Baseline 1 algorithm, we derived a new algorithm from the proposed method by limiting the number of degrees to only two. In other words, we only use two types of degrees: $R_{\approx}$ and $R_{>}$. The other degrees, $R_{\gg}, R_{\ggg}$, and $R_{\gggg}$,are merged into $R_{>}$. And the derived algorithm does not distinguish among the experts. This derived algorithm can be seen as a kind of learning-to-rank algorithm with a pairwise approach.

**Baseline 2.** In general, we cannot know the boundaries between the degrees in advance. In the Baseline 2 algorithm, we assume that the sizes of the degrees are all the same, and there is no distinction among the experts. This algorithm is virtually identical to SVM with $\epsilon$-insensitive error.

First, we investigated whether our method can estimate the correct boundaries ($b^*_{1,0}$ etc.) that may vary according to each expert. We generated 500 sample solution pairs by the above-mentioned method, and assign 50%, 30% and 20% of the pairs to Expert1, Expert2 and Expert3 respectively. This allowed us to investigate our method in a heterogeneous setting. Figures 1, 2 and 3 show the differences between the objective values of a sample solution pair, and to which degree (such as $R_{\approx}, R_{>}$, etc.) each sample solution pair belongs, using the proposed method, Baseline 1 and Baseline 2 respectively. The horizontal axis represents the difference between the objective values of a sample solution pair with respect to the estimated $f_{\boldsymbol{w}}$. For the vertical axis, the vertical coordinate is set at random within each expert. These figures also show the correct and estimated boundaries. Note that the boundaries estimated by Baselines 1 and 2 span all the experts because these algorithms do not distinguish among the experts. For Baseline 1, there is only one estimated boundary because this algorithm can deal with only two degrees, which corresponds to one boundary between them. These figures show that our method can precisely estimate the boundaries, even though each expert would set them differently. For Baselines 1 and 2, many sample points are located outside of the correct boundaries because the estimation of $\boldsymbol{w}$ is not sufficiently accurate.

Next we evaluated these algorithms by using the relative error of $\boldsymbol{w}$ defined as:

$$RelativeError(\boldsymbol{w}) \equiv \frac{1}{K} \sum_{1 \le k \le K} \frac{|w^*_k - w_k|}{w^*_k}, \qquad (25)$$

where $w^*_k$ is the correct weight for the objective term $k$. Figure 4 shows the relative errors of each algorithm with sev-

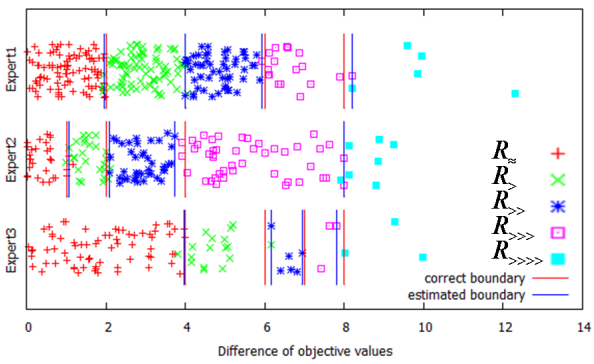


Figure 1: The estimated boundaries by Proposed method

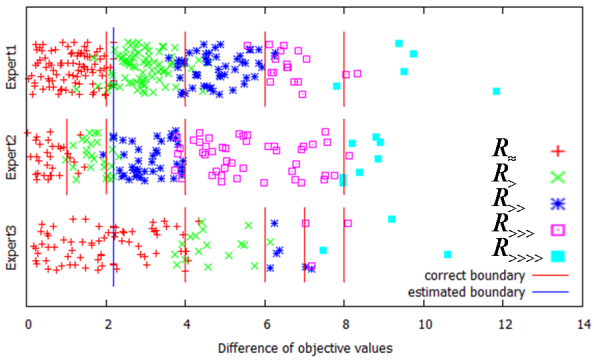


Figure 2: The estimated boundary by Baseline 1

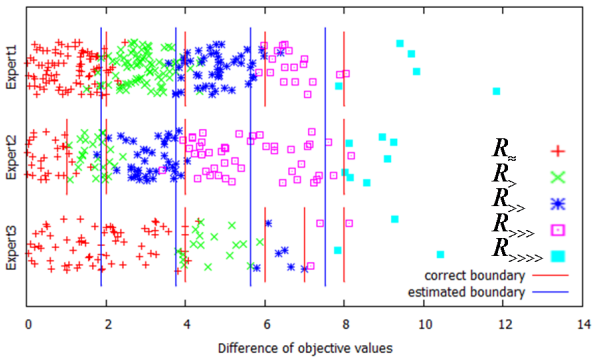


Figure 3: The estimated boundaries by Baseline 2

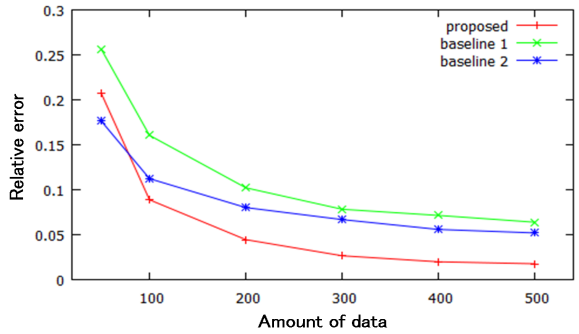


Figure 4: Relative errors of the algorithms

eral amounts of training data (as measured by the number of solution pairs used for estimating the weights). For each amount of training data, we ran 100 trials with different random seeds, and averaged them to plot the graph. While the vertical axis represents the relative error, the horizontal axis represents the amount of training data.

Figure 4 shows that the proposed algorithm can produce more accurate estimates for every amount of training data compared to Baseline 1. The essential difference between the proposed method and Baseline 1 is that the proposed algorithm can appropriately deal with all of the boundaries to estimate the weights $\boldsymbol{w}$, while Baseline 1 cannot. In other words, the proposed algorithm can extract much more information from the experts' evaluations.

The proposed algorithm is also superior to Baseline 2 except for the data set with 50 ratings. When the amount of data is 500 ratings, the relative error of the proposed method (1.76%) is about one-third of that of Baseline 2 (5.20%). When there are 50 ratings, Baseline 2 is better than the proposed algorithm. This is because the proposed method has to divide the training data into three groups, which is the number of experts, and estimates the boundaries for each expert independently. As a result, the actual amount of training data becomes too small to accurately estimate the boundaries. The amount of training data (50 ratings) is too small to estimate the 12 boundaries (4 boundaries $\times$ 3 experts). This should be regarded as a data issue rather than an algorithmic flaw of the proposed method.

## Conclusions and Future Work

We are studying a method to reformulate a machine learning problem as a mathematical programming problem. Thanks to the strong modeling capabilities of mathematical programming, we showed that even a quadratic function and regularization functionality can be described in an LP. Utilizing these powerful capabilities, we proposed a novel method to determine the objective function from the experts' evaluations. Our method can properly handle qualitative and subjective data, which is quite common for human expert evaluations. The experimental results showed that the proposed method is superior to two simpler algorithms.

We also showed that basis function selection can be implemented in our formulation. To assess this capability, however, we need to enumerate appropriate candidate basis functions for each objective term, which should depend on the applications. This is another research issue, but we leave this as future work with real application data.

While the machine learning and mathematical programming fields are quite different, our research implies that the techniques developed in the mathematical programming community could be strong tools for solving some machine learning problems. Various problems that have been regarded as quite difficult in the machine learning community might be more easily solved using techniques from the mathematical programming community. Our work represents a new research direction for both of machine learning and mathematical programming research.

## References


Bennett, K. P., and Parrado-Hernández, E. 2006. The interplay of optimization and machine learning research. *The Journal of Machine Learning Research* 7:1265–1281.

Cao, Y.; Xu, J.; Liu, T.-Y.; Li, H.; Huang, Y.; and Hon, H.-W. 2006. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 186–193. ACM.

Dantzig, G. B. 1998. *Linear programming and extensions*. Princeton university press.

Forman, E. H., and Gass, S. I. 2001. The analytic hierarchy process: An exposition. *Operations Research* 49(4):469–486.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 133–142. ACM.

Johnson, E. L.; Nemhauser, G. L.; and Savelsbergh, M. W. 2000. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing* 12(1):2–23.

Likert, R. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

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

Steinwart, I., and Christmann, A. 2009. Sparsity of svms that use the epsilon-insensitive loss. In *Advances in Neural Information Processing Systems*, 1569–1576.

Stevens, S. S. 1946. On the theory of scales of measurement. *Science* 103(2684):667–680.