1-2015

Optimizing pred(25) Is NP-Hard

Martine Ceberio
University of Texas at El Paso, mceberio@utep.edu

Olga Kosheleva
University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-15-06

Optimizing pred(25) Is NP-Hard

Martine Ceberio, Olga Kosheleva, and Vladik Kreinovich

University of Texas at El Paso, El Paso, TX 79968, USA
{mceberio,olgak,vladik}@utep.edu

Abstract. Usually, in data processing, to find the parameters of the models that best fits the data, people use the Least Squares method. One of the advantages of this method is that for linear models, it leads to an easy-to-solve system of linear equations. A limitation of this method is that even a single outlier can ruin the corresponding estimates; thus, more robust methods are needed. In particular, in software engineering, often, a more robust pred(25) method is used, in which we maximize the number of cases in which the model's prediction is within the 25% range of the observations. In this paper, we show that even for linear models, pred(25) parameter estimation is NP-hard.

1 Formulation of the Problem

Need to estimate parameters of models. In many practical situations, we know that a quantity y depends on the quantities $x_1, \ldots, x_n$, and we know the general type of this dependence. In precise terms, this means that we know a family of functions $f(c_1, \ldots, c_p, x_1, \ldots, x_n)$ characterized by parameters c_i, and we know that the actual dependence corresponds to one of these functions.

For example, we may know that the dependence is linear; in this cases, the corresponding family takes the form

$$f(c_1, \ldots, c_n, c_{n+1}, x_1, \ldots, x_n) = c_{n+1} + \sum_{i=1}^{n} c_i \cdot x_i.$$

In general, we know the type of the dependence, but we do not know the actual values of the parameters. These values can only be determined from the measurements and observations, when we observe the values x_j and the corresponding value y. Measurement and observations are always approximate, so we end up with tuples $(x_{1k}, \ldots, x_{nk}, y_k)$, $1 \le k \le K$, for which $y_k \approx f(c_1, \ldots, c_p, x_{1k}, \ldots, x_{nk})$ for all k. We need to estimate the parameters $c_1, \ldots, c_p$ based on these measurement results.

Least Squares: traditional way of estimating parameters of models. In most practical situations, the Least Squares method is used to estimate the desired parameters. In this method, we select the values c_i for which the sum of the squares of the approximation errors is the smallest possible:

$$\sum_k (y_k - f(c_1, \ldots, c_p, x_{1k}, \ldots, x_{nk}))^2 \to \min_{c_1, \ldots, c_p}.$$

One of advantages of this approach is that, when the model $f(c_1, \ldots, c_p, x_1, \ldots, x_n)$ linearly depends on the parameters c_i, the sum of squares is a quadratic function of c_i. Thus, when we apply the usual criterion for the minimum – differentiate the sum with respect to each variable x_i and equate all the resulting partial derivatives to 0 – we get a system of linear equations, from which we can easily find all the unknown $c_1, \ldots, c_p$.

Least Squares is not always the optimal way of estimating the parameters. The Least Squares approach known to be optimal for the case when all the approximation errors $y_k - f(c_1, \ldots, c_p, x_{1k}, \ldots, x_{nk})$ are independent and all distributed according to the same normal distribution. In practice, however, we often have outliers – e.g., values corresponding to the malfunction of a measuring instrument – and in the presence of even a single outlier, the Least Squares method can give very wrong results.

Let us illustrate this on the simplified example, when y does not depend on any variables x_i at all, i.e., when $y = c$ for some unknown constant c. In this case, we need to estimate the value c based on the observations $y_1, \ldots, y_K$. For this problem, the Least Squares method takes the form $\sum\limits_{k=1}^{K} (y_k - c)^2 \to \min$. Differentiating the sum with respect to the unknown c and equating the derivative to 0, we conclude that $c = \dfrac{y_1 + \ldots + y_K}{K}$.

This formula works well if all the values y_i are approximately equal to c. For example, if the actual value of c is 0, and $|y_i| \le 0.1$, we get an estimate $|c| \le 0.1$. However, if out of 100 measurements y_i, one of an outlier equal to 1000, the estimate becomes close to 10 – and thus, far away from the actual value 0.

To take care of such situations, we need estimates which do not change as much in the presence of possible outliers. Such methods are called *robust* [2].

pred(25) as an example of a robust estimate. One of the possible robust estimates consists of selecting a percentage α and selecting the values of the parameters for which the number of observations for which the prediction is within α% from the observed value is the largest possible. In other words, each prediction is formulated as a constraint, and we look for parameters that maximize the number of satisfied constraint. This technique is known as pred(α).

This method is especially widely used in software engineering, e.g., for estimating how well different models can predict the overall software effort and/or the number of bugs. In software engineering, this method is most frequently applied as pred(25), for $\alpha = 25$; see, e.g., [1, 3].

Problem. In contrast to the Least Squares approach, for which the usual calculus ideas lead to an efficient optimization algorithm, no such easy solution is known for pred(25) estimates; all known algorithms for this estimation are rather time-consuming. A natural question arises: is this because we have not yet found a feasible algorithm for computing these estimates, or is this estimation problem really hard?

What we prove in this paper. In this paper, we prove that even for a linear model with no free term c_{n+1}, pred(25) estimation – as well as pred(α) estimation

for any $\alpha > 0$ – is an NP-hard problem. In plain terms, this means that this problem is indeed inherently hard.

2 Main Result and Its Proof

Definition 1. *Let $\alpha \in (0,1)$ be a rational number. By a* linear pred(α)-estimation problem*, we means the following problem:*

- Given: *an integer n, K rational-valued tuples $(x_{1k}, \ldots, x_{nk}, y_k)$, $1 \le k \le K$, and an integer $M < K$;*
- Check: *whether there exist parameters $c_1, \ldots, c_n$ for which in at least M cases k, we have*

$$\left| y_k - \sum_{i=1}^{n} c_i \cdot x_{ik} \right| \le \alpha \cdot \left| \sum_{i=1}^{n} c_i \cdot x_{ik} \right|.$$

Proposition 1. *For every α, the linear pred(α)-estimation problem is NP-hard.*

Proof. To prove this result, we will reduce, to this problem, a known NP-hard problem of checking whether a set of integer weights $s_1, \ldots, s_m$ can be divided into two parts of equal overall weight, i.e., whether there exist integers $y_j \in \{-1, 1\}$ for which $\sum\limits_{j=1}^{m} y_j \cdot s_j = 0$; see, e.g., [4].

In the reduced problem, we will have $n = m + 1$, with $n = m + 1$ unknown coefficients $c_1, \ldots, c_m, c_{m+1}$. The parameters c_i will correspond to the values y_i, and c_{m+1} is equal to 1. We will build tuples corresponding to equations $y_i = 1$ and $y_i = -1$ for $i \le m$, to $c_{m+1} = 1$, and to the equation $c_{m+1} + \sum\limits_{i=1}^{m} y_i \cdot s_i = 1$.

To each equation of the type $y_i = 1$ or $c_{m+1} = 1$, we put into correspondence the following two tuples:

- In the first tuple, $x_{ik} = 1 + \varepsilon$, $x_{jk} = 0$ for all $j \ne i$, and $y_k = 1$. The resulting linear term has the form $c_i \cdot (1 + \varepsilon)$ and thus, the corresponding inequality takes the form $1 - \varepsilon \le (1 + \varepsilon) \cdot c_i \le 1 + \varepsilon$, i.e., equivalently, the form $\dfrac{1 - \varepsilon}{1 + \varepsilon} \le c_i \le 1$.
- In the second tuple, $x_{ik} = 1 - \varepsilon$, $x_{jk} = 0$ for all $j \ne i$, and $y_k = 1$. The resulting linear term has the form $c_i \cdot (1 - \varepsilon)$ and thus, the corresponding inequality takes the form $1 - \varepsilon \le (1 - \varepsilon) \cdot c_i \le 1 + \varepsilon$, i.e., equivalently, the form $1 \le c_i \le \dfrac{1 + \varepsilon}{1 - \varepsilon}$.

It should be mentioned that the only value c_i that satisfies both inequalities is the value $c_i = 1$.

Similarly, to each equation of the type $y_i = -1$, we put into correspondence following two tuples.

– In the first tuple, $x_{ik} = 1 + \varepsilon$, $x_{jk} = 0$ for all $j \neq i$, and $y_k = -1$. The resulting linear term has the form $c_i \cdot (1 + \varepsilon)$ and thus, the corresponding inequality takes the form $-1 - \varepsilon \leq (1 + \varepsilon) \cdot c_i \leq -1 - \varepsilon$, i.e., equivalently, the form $-1 \leq c_i \leq -\dfrac{1 - \varepsilon}{1 + \varepsilon}$.
– In the second tuple, $x_{ik} = 1 - \varepsilon$, $x_{jk} = 0$ for all $j \neq i$, and $y_k = -1$. The resulting linear term has the form $c_i \cdot (1 - \varepsilon)$ and thus, the corresponding inequality takes the form $-1 - \varepsilon \leq (1 - \varepsilon) \cdot c_i \leq -1 + \varepsilon$, i.e., equivalently, the form $-\dfrac{1 + \varepsilon}{1 - \varepsilon} \leq c_i \leq -1$.

Here also, the only value c_i that satisfies both inequalities is the value $c_i = -1$.

Finally, to the equation $c_{m+1} + \sum\limits_{j=1}^{m} y_j \cdot s_j = 1$, we put into correspondence the following two tuples. In both tuples, $y_k = 1$.

– In the first tuple, $x_{ik} = (1 + \varepsilon) \cdot s_i$, and $x_{m+1,k} = 1 + \varepsilon$. The corresponding linear term has the form $(1 + \varepsilon) \cdot \left(\sum\limits_{i=1}^{m} c_i \cdot s_i + c_{m+1} \right)$, and thus, the corresponding inequality takes the form

$$1 - \varepsilon \leq (1 + \varepsilon) \cdot \left(\sum_{i=1}^{m} c_i \cdot s_i + c_{m+1} \right) \leq 1 + \varepsilon,$$

i.e., equivalently,

$$\frac{1 - \varepsilon}{1 + \varepsilon} \leq \sum_{i=1}^{m} c_i \cdot s_i + c_{m+1} \leq 1.$$

– In the second tuple, $x_{ik} = (1 - \varepsilon) \cdot s_i$, and $x_{m+1,k} = 1 - \varepsilon$. The corresponding linear term has the form $(1 - \varepsilon) \cdot \left(\sum\limits_{i=1}^{m} c_i \cdot s_i + c_{m+1} \right)$, and thus, the corresponding inequality takes the form

$$1 - \varepsilon \leq (1 - \varepsilon) \cdot \left(\sum_{i=1}^{m} c_i \cdot s_i + c_{m+1} \right) \leq 1 + \varepsilon,$$

i.e., equivalently,

$$1 \leq \sum_{i=1}^{m} c_i \cdot s_i + c_{m+1} \leq \frac{1 + \varepsilon}{1 - \varepsilon}.$$

Here, both inequalities are satisfied if and only if $\sum\limits_{i=1}^{m} c_i \cdot s_i + c_{m+1} = 1$.

Overall, we have $2m + 2$ pairs, i.e., $4m + 4$ tuples. If for the given values $s_1, \ldots, s_m$, the original NP-hard problem has a solution y_i, then we can take $c_i = y_i$, $c_{m+1} = 1$, and thus satisfy $M \stackrel{\text{def}}{=} 2m + 4$ inequalities. Let us show that, vice versa, if at least $2m+4$ inequalities are satisfied, this means that the original problem has a solution.

Indeed, for every i, each of the two inequalities corresponding to $y_i = 1$ implies that $c_i > 0$ while each of the two inequalities corresponding to $y_i = -1$ implies that $c_i < 0$. Thus, these inequalities incompatible, which means that for every i, at most two inequalities can be satisfied. If for some i, fewer than two inequalities are satisfied, then even when for every $j \neq i$, we have two, and all four remaining inequalities are satisfied, we will still have fewer than $2m + 4$ satisfied inequalities. This means that if $2m+4$ inequalities are satisfied, then for every i, two inequalities are satisfied – and thus, either $c_i = 1$ or $c_i = -1$. Now, the four additional inequalities also have to be satisfied, so we have $c_{m+1} = 1$, and $\sum_{i=1}^{m} c_i \cdot s_i + c_{m+1} = 1$, hence $\sum_{i=1}^{m} c_i \cdot s_i = 0$. The reduction is proven, and thus our problem is indeed NP-hard.

Comment. In this proof, we consider situations in which about half of the inequalities are satisfied. We may want to restrict ourselves to situations in which a certain proportion of inequality should be satisfied – e.g., 90% or 99%. With such a restriction, the problem remains NP-hard.

To prove this, it is sufficient to consider a similar reduction, in which:

- instead of single pair of tuples corresponding to $c_{m+1} = 1$ we have N identical pairs (for a sufficiently large N), and similarly,
- instead of a single pair corresponding to the equation $\sum_{j=1}^{m} y_j \cdot s_j = 0$, we have N such identical pairs.

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

References

1. S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin/Cummings, Menlo Park, California, 1986.
2. P. J. Huber, *Robust Statistics*, Wiley, Hoboken, New Jersey, 2004.
3. E. Mendes, *Cost Estimation Techniques for Web Projects*, IGI Publ., Hershey, Pennsylvania, 2007.
4. C. H. Papadimitriou, *Computational Complexity*, Addison Wesley, San Diego, 1994.