IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 22, 2021, accepted January 26, 2021, date of publication February 3, 2021, date of current version February 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3056710

Divergence Family Contribution to Data Evaluation in Blockchain via Alpha-EM and Log-EM Algorithms

YASUO MATSUYAMA, (Life Fellow, IEEE)
Research Institute of Science and Engineering, Waseda University, Tokyo 169-8555, Japan
e-mail: yasuo2@waseda.jp

ABSTRACT This study interrelates three adjacent topics in data evaluation. The first is the establishment of a relationship between Bregman divergence and probabilistic alpha-divergence. In particular, we demonstrate that square-root-order probability normalization enables the unification of these two divergence families. This yields a new alpha-divergence, which can be used to jointly derive the alpha-EM algorithm (alpha-expectation-maximization algorithm) and the traditional log-EM algorithm. The second topic is the application of the alpha-EM algorithm in the evaluation of graders scoring raw data over a network. We estimate multinomial mixture distributions in this evaluation problem. We note that the convergence speed of the alpha-EM algorithm is significantly higher than that of the log-EM algorithm. Finally, the third topic is the use of this increase in convergence speed to assign the winning evaluator and miner in a blockchain environment. This is achieved by proof-of-review using evaluation scores, which is a class of proof-of-stake. In the second and third topics, we select terminology from wine tasting for brevity in the exposition. However, this formulation can be applied to a broader class of data in a network environment comprising blockchains.

INDEX TERMS Alpha-EM algorithm, blockchain, data evaluation, divergence, dynamic information bank, log-EM algorithm, multinomial mixture distribution, proof-of-review, proof-of-stake.

I. INTRODUCTION

Measuring the divergence or distance[1] between two probabilities is fundamental in data evaluation. In this paper, we present a path from a new divergence measure to blockchain applications. As a highlighting relay point, by utilizing this measure, we address the trust evaluation of graders who score raw data.

Since the early days of the theory of information and communication, a wide range of distance measures have been developed: average mutual information [1], Kullback–Leibler divergence [2], and more general divergences [3]–[5]. There is a hierarchy in these quantities according to the level of generalization. Generalization is essential for the deepening of the theory. However, in applications to actual data, the appropriate level of abstraction should be determined. In this study, we first establish a relationship between Bregman divergence [6] and α-divergence [7], [8]. Here, we present a skewing technique for the probability space by geometric methods, such as taking the square root of probabilities. Thus, we obtain a new α-divergence that is different from that used to derive the α-EM algorithm (alpha-expectation-maximization algorithm) [8], [9]. This measure is termed Type-II α-divergence. It has an important property whereby the traditional log-EM algorithm [10] and the α-EM algorithm [8] can be jointly derived.

The EM family has a wide range of applications in statistical data processing and machine learning. Convergence speed is essential in every case. The α-EM algorithm has theoretically proven faster convergence than the log-EM algorithm [8]. Accordingly, we use the α-EM algorithm to evaluate graders who score raw data collected in a network environment. We employ a statistical model by mixing multinomial distributions [11], [12] using the α-EM algorithm. To provide a concrete instance, we reformulate this as a sommelier qualification problem. The raw data for sommeliers are wines. However, the type of these raw data may change upon request by a data collector or bookmaker, and therefore, the method applies to general data over a network. We present various strategies for achieving high convergence speed.

Following these considerations regarding convergence speed gains, we apply the α-EM algorithm in a blockchain environment. Convergence speed becomes an additional factor in determining the winner or the most trustable eval-

The associate editor coordinating the review of this manuscript and approving it for publication was Yassine Maleh.

[1]Strictly speaking, the distance must satisfy the symmetry and triangle inequality. In this paper, we use the word ''distance'' to describe the dissimilarity for convenience.

uator, whose task is to review all sommelier reports. We use this mechanism for consensus building among miners. The strategy is a type of proof-of-stake (PoS) [13]. As the review of sommelier reports is essential, we term this method *proof-of-review* (PoR). Only the miner who receives a winning token can generate a new block in the blockchain. All stakeholders contributing to this trust decision receive rewards such as crypto-assets. This formulation can be easily recast to other data sent from IoT devices under the framework of this *dynamic information bank*.

Thus, the contributions of this study are the following:

(1) Unification of Bregman divergence and α-divergence by skewing probability spaces: Type-II α-divergence contributes to the joint derivation of the α-EM and log-EM algorithms.
(2) Theory and experiments on the α-EM algorithm as an evaluator that qualifies graders: We present this trust evaluation by a mixture of multinomial distributions. As a highlight, we demonstrate various strategies for exploiting the speed of the α-EM algorithm supported by experiments.
(3) Method for exploiting computing speed in PoR in a blockchain environment: We present a method for using the computing speed of the α-EM algorithm to set up a legitimate mining right in the blockchain.

A brief preliminary partial report on the above appears in [14]. However, by previewing Fig. 1 in Section IV-A, the new contributions of this study can be comprehensively noted.

II. MATHEMATICAL RELATIONSHIPS BETWEEN BREGMAN DIVERGENCE AND α-DIVERGENCE

A. BREGMAN DIVERGENCE

Bregman divergence [6] quantifies the distance between two vector points $\boldsymbol{u}$ and $\boldsymbol{v}$ by using a convex function f and its tangent:

$$b_f(\boldsymbol{u}, \boldsymbol{v}) = f(\boldsymbol{u}) - f(\boldsymbol{v}) - \nabla f(\boldsymbol{v})^T(\boldsymbol{u} - \boldsymbol{v}). \tag{1}$$

Here, ∇ denotes the gradient. As we consider information divergence based on probabilistic quantities, $\boldsymbol{u}$ and $\boldsymbol{v}$ are scalars, that is, $u = p(\boldsymbol{x})$ and $v = q(\boldsymbol{x})$. As these quantities are probabilities, we have the following constraints. For probability densities, the constraints are

$$\int_{\mathcal{X}} p(\boldsymbol{x})d\boldsymbol{x} = \int_{\mathcal{X}} q(\boldsymbol{x})d\boldsymbol{x} = 1. \tag{2}$$

In the case of probability masses, the integrals become sums:

$$\sum_{n=1}^{N} p(\boldsymbol{x}_n) = \sum_{n=1}^{N} q(\boldsymbol{x}_n) = 1. \tag{3}$$

The constraints (2) and (3) cause the geometric skewing of probability spaces, so that Bregman divergence and α-divergence coincide.

B. PROBABILITY SPACE SKEWING FOR TYPE-I α-DIVERGENCE

We derive α-divergence from Bregman divergence because of the general expression (1) for the latter. Let $f^{(\alpha}(r)$ be a convex function such that

$$\begin{aligned} f^{(\alpha)}(r) &\stackrel{def}{=} L^{(\alpha)}(r) \times M^{(\alpha)}(r) \\ &\stackrel{def}{=} \frac{2}{1+\alpha}(r^{\frac{1+\alpha}{2}} - 1) \times \frac{2}{1-\alpha} r^{\frac{1-\alpha}{2}}. \end{aligned}$$

Here,

$$L^{(\alpha)}(r) = \frac{2}{1+\alpha}(r^{\frac{1+\alpha}{2}} - 1), \tag{4}$$

with $r > 0$ and $\alpha < 1$, is the α-logarithm [8], [9]. The limit of $L^{(\alpha)}$ as $\alpha \to -1$ is the usual logarithm by L'Hôspital's rule, whereas if $\alpha = 1$, we obtain a linear function.

Let the Bregman divergence between p and q using the convex function $f^{(\alpha)}(q)$ be $b_f^{(\alpha)}(p, q)$:

$$b_f^{(\alpha)}(p, q) = f^{(\alpha)}(p) - f^{(\alpha)}(q) - \nabla f^{(\alpha)}(q)(p - q).$$

Then, we have the following relationship in a skewed probability space using $q^{\frac{1+\alpha}{2}}$:

$$\int_{\mathcal{X}} q^{\frac{1+\alpha}{2}} b_f^{(\alpha)}(p, q)d\boldsymbol{x} = \frac{4}{1-\alpha^2}\{1 - \int_{\mathcal{X}} p(\frac{q}{p})^{\frac{1+\alpha}{2}} d\boldsymbol{x}\}. \tag{5}$$

We denote the left-hand side of (5) by $B_f^{(\alpha)}(p||q)$. The right-hand side can be identified as the α-divergence $D_f^{(\alpha)}(p||q)$, which was used to derive the α-EM algorithm [8]. As we present a different α-divergence in the next subsection, we term this measure Type-I α-divergence. Then, by changing p and q, we have the following skewed dualities:

$$\begin{aligned} D_I^{(\alpha)}(p||q) &\stackrel{def}{=} D_f^{(\alpha)}(p||q) = B_f^{(\alpha)}(p||q) \\ &= B_f^{(-\alpha)}(q||p) = D_f^{(-\alpha)}(q||p). \end{aligned} \tag{6}$$

The same equations can be obtained by skewing $b_f^{(-\alpha)}(q, p)$ using $p^{\frac{1-\alpha}{2}}$.

C. PROBABILITY SKEWING FOR TYPE-II α-DIVERGENCE

1) DIRECT APPROACH

We can obtain another class of α-divergences. The logarithmic version of this measure as a spectral distance, termed the Itakura–Saito distance, was used in speech processing [15], [16] because of the Pythagorean theorem in spectral estimation. However, we cannot use this quantity as a probabilistic distance because of the presence of a constant term, the integral of which over the probability space is infinite. However, Type-II α-divergence can be derived by the proposed Bregman divergence approach using probability skewing as follows.

Let the convex function for the Bregman divergence be $g(r) = -\log r$ instead of $f(\boldsymbol{u})$ in (1). Then, we have

$$b_g(u, v) = \frac{u}{v} - \log\frac{u}{v} - 1.$$

We now stretch the probabilities by $u = q^\beta$ and $v = p^\beta$. This skewing of the probability space leads to the following equation:

$$\frac{1}{\beta^2}\int_{\mathcal{X}} p b_g(p^\beta, q^\beta)d\boldsymbol{x} = \frac{1}{\beta^2}\int_{\mathcal{X}} p\{(\frac{q}{p})^\beta - \log(\frac{q}{p})^\beta - 1\}d\boldsymbol{x}. \tag{7}$$

We change the parameter by $\beta = \frac{1-\alpha}{2}$, and denote the left-hand side of (7) by $B_g^{(\alpha)}(p||q)$. Then, we have the

following equations that define Type-II α-divergence:

$$B_g^{(\alpha)}(p||q) = \begin{cases} \left(\frac{2}{1-\alpha}\right)^2 \int_{\mathcal{X}} p \left\{ \left(\frac{q}{p}\right)^{\frac{1-\alpha}{2}} - \log\left(\frac{q}{p}\right)^{\frac{1-\alpha}{2}} - 1 \right\} d\boldsymbol{x}, & (\alpha \neq 1) \\ \frac{1}{2} \int_{\mathcal{X}} p \left(\log \frac{q}{p}\right)^2 d\boldsymbol{x}, & (\alpha = 1) \\ K(p||q), & (\alpha = -1) \end{cases}$$
$$\stackrel{def}{=} D_{II}^{(\alpha)}(p||q). \tag{8}$$

Here, the case $\alpha = 1$ is obtained by L'Hôspital's rule. Letting $\alpha = -1$ on the right-hand side of (8) yields the Kullback–Libler divergence [2]:

$$K(p||q) = \int_{\mathcal{X}} p \log \frac{p}{q} d\boldsymbol{x}. \tag{9}$$

In (8), q is the target probability to be modeled by p. This choice is compatible with the Itakura–Saito distance for inverse filter matching [16].

2) TWO-PARAMETER APPROACH

We can obtain (8) using a two-parameter kernel, which is related to the discrete kernel that appears in [17]. Here, the kernel for $r \in (0, \infty)$ is

$$h^{(\alpha,\beta)}(r) \stackrel{def}{=} \frac{2}{\alpha-\beta}\{\frac{2}{1-\alpha}(r - r^{\frac{1-\alpha}{2}}) - \frac{2}{1-\beta}(r - r^{\frac{1-\beta}{2}})\}. \tag{10}$$

By using this kernel, we define a two-parameter divergence as follows:

$$D_h^{(\alpha,\beta)}(p||q) \stackrel{def}{=} \int_{\mathcal{X}} q h^{(\alpha,\beta)}(p/q) d\boldsymbol{x}. \tag{11}$$

Then, we have

$$D_h^{(\alpha,1)}(p||q) = B_g^{(\alpha)}(p||q) = D_{II}^{(\alpha)}(p||q). \tag{12}$$

Here, by L'Hôspital's rule, we obtain the case $\beta = 1$. Thus, Bregman divergence and α-divergence have a strong relationship [18]. However, the probability skewing by $p^{\frac{1-\alpha}{2}}$ and $q^{\frac{1-\alpha}{2}}$ was required for Type-I α-divergence in (6). For Type-II α-divergence in (8), we applied probability skewing by ratio-stretching $(\frac{q}{p})^{\beta} = (\frac{q}{p})^{\frac{1-\alpha}{2}}$.

We conclude this section by observing the following relationships:

$$\begin{aligned} D_I^{(-1)}(p||q) &= D_{II}^{(-1)}(p||q) = K(p||q) \\ &= \int_{\mathcal{X}} p \log(1/q) d\boldsymbol{x} - \int_{\mathcal{X}} p \log(1/p) d\boldsymbol{x} \\ &\stackrel{def}{=} C(p, q) - H(p). \end{aligned} \tag{13}$$

Here, the first term, $C(p, q)$, is the cross-entropy, and the second term, $H(p) = C(p, p)$, is the entropy. We note that, in deep-learning techniques as in [19], the cross entropy is used to examine the convergence of learning. Then, $p = p_{teacher}$ corresponds to the teacher signal, and $q = q_{output}$ is the model output.

III. RELATIONSHIPS OF TYPE-II α-DIVERGENCE with log-EM ALGORITHM AND α-EM ALGORITHM

As seen above, there are certain relationships between Bregman divergence and α-divergence. However, it is important to determine their usefulness in practice. Herein, we demonstrate that Type-II α-divergence can be used to jointly derive the α-EM [8] and log-EM algorithms [10].

A. ANOTHER EXPRESSION OF TYPE-II α-DIVERGENCE

We first derive the relationship of Type-II α-divergence with Type-I α-divergence:

$$D_{II}^{(\alpha)}(p||q) = \frac{2}{1-\alpha} K(p||q) - \frac{1+\alpha}{1-\alpha} D_I^{(-\alpha)}(p||q). \tag{14}$$

This can be obtained directly from (8) or by computing (12) using the definitions in (10) and (11). The derivation is rather lengthy and is provided in Appendix A.

Hereafter, we use conventional probabilistic notations: $p(\boldsymbol{x}|\psi)$, $p(\boldsymbol{x}|\varphi)$, $p(\boldsymbol{y}|\psi)$, $p(\boldsymbol{y}|\varphi)$, and $p(\boldsymbol{x}|\boldsymbol{y}, \psi)$, and $p(\boldsymbol{x}|\boldsymbol{y}, \varphi)$. Here, ψ and φ are parameters that define the forms of the probability functions. For notational simplicity, the symbols $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ below are also used for the corresponding random variables.

The random variable $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$ is complete data, comprising incomplete data $\boldsymbol{y}$ and missing data $\boldsymbol{z}$. By using this relationship, we have

$$p(\boldsymbol{x}|\boldsymbol{y}, \psi) = \frac{p(\boldsymbol{x}|\psi)}{p(\boldsymbol{y}|\psi)} \tag{15}$$

because the incomplete data $\boldsymbol{y}$ are partial information of the complete data $\boldsymbol{x}$. Here, the joint probability density $p(\boldsymbol{x}, \boldsymbol{y}|\psi)$ is denoted by $p(\boldsymbol{x}|\psi)$.

B. JOINT DERIVATION OF log-EM AND α-EM

We use Type-II α-divergence to compute the distance between two conditional probability densities $p(\boldsymbol{x}|\boldsymbol{y}, \varphi)$ and $p(\boldsymbol{x}|\boldsymbol{y}, \psi)$. As the notation of this divergence is rather cumbersome, we abbreviate it as

$$D_{II}^{(\alpha)}(p(\boldsymbol{x}|\boldsymbol{y}, \varphi)||p(\boldsymbol{x}|\boldsymbol{y}, \psi)) \stackrel{def}{=} D_{II}^{(\alpha)}(\varphi||\psi).$$

Similarly, Type-I α-divergence is abbreviated as $D_I^{(\alpha)}(\varphi||\psi)$.

We begin with the computation of $D_{II}^{(\alpha)}(\varphi||\psi)$, and by (14), we have

$$\begin{aligned} &D_{II}^{(\alpha)}(\varphi||\psi) \\ &= \frac{2}{1-\alpha} L_{\boldsymbol{y}}^{(-1)}(\psi|\varphi) - \frac{2}{1-\alpha}\left\{\frac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\right\}^{-\frac{1-\alpha}{2}} L_{\boldsymbol{y}}^{(-\alpha)}(\psi|\varphi) \\ &- \frac{2}{1-\alpha} Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\psi|\varphi) + \frac{2}{1-\alpha}\left\{\frac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\right\}^{-\frac{1-\alpha}{2}} Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-\alpha)}(\psi|\varphi). \end{aligned} \tag{16}$$

Here, the probabilistic quantities are as follows:

$$L_{\boldsymbol{y}}^{(-\alpha)}(\psi|\varphi) = L^{(-\alpha)}(\frac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}), \tag{17}$$

$$Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-\alpha)}(\psi|\varphi) = \int_{\mathcal{X}(\boldsymbol{y})} p(\boldsymbol{x}|\boldsymbol{y}, \varphi) L_{\boldsymbol{x}}^{(-\alpha)}(\psi|\varphi) d\boldsymbol{x}. \tag{18}$$

The region of integration $\mathcal{X}(\boldsymbol{y})$ is a subset of $\mathcal{X}$ that generates the incomplete data $\boldsymbol{y}$. The derivation of (16) is presented in Appendix B.

We now use the relationship of Type-II and Type-I α-divergence. By using (14) and (16) and changing $-\alpha$ to

α, we have

$$L_{\boldsymbol{y}}^{(-1)}(\psi|\varphi) - Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\psi|\varphi) - K(\varphi||\psi) = \{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\}^{-\frac{1+\alpha}{2}} \left[L_{\boldsymbol{y}}^{(\alpha)}(\psi|\varphi) - Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(\alpha)}(\psi|\varphi) - \tfrac{1-\alpha}{2}\{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\}^{\frac{1+\alpha}{2}} D_I^{(\alpha)}(\varphi||\psi) \right]. \quad (19)$$

By L'Hôspital's rule, we have $L^{(-1)}(r) = \log r$. Therefore, the left-hand side of (19) is equal to zero. This is the basic equation [8], [9] of the log-EM algorithm:

$$L_{\boldsymbol{y}}^{(-1)}(\psi|\varphi) = Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\psi|\varphi) + K(\varphi||\psi). \quad (20)$$

The log-EM algorithm is derived by appropriately interpreting this equation.

log-EM algorithm: To increase the incomplete-data likelihood $p(\boldsymbol{y})$, the incomplete-data log-likelihood ratio $L_{\boldsymbol{y}}^{(-1)}(\psi|\varphi)$ should be positive. As the Kullback–Leibler divergence $K(\varphi||\psi)$ is positive before convergence, the positivity of $L_{\boldsymbol{y}}^{(-1)}(\psi|\varphi)$ is achieved by maximizing the Q-function $Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\psi|\varphi)$ (and thus making it positive as well). This continues by replacing φ with ψ until convergence is attained.

We now have the following equation from (19) and (20):

$$L_{\boldsymbol{y}}^{(\alpha)}(\psi|\varphi) = Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(\alpha)}(\psi|\varphi) + \tfrac{1-\alpha}{2}\{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\}^{\frac{1+\alpha}{2}} D_I^{(\alpha)}(\varphi||\psi). \quad (21)$$

This is the basic equation for the α-EM algorithm.

α-EM algorithm: To increase the incomplete-data likelihood $p(\boldsymbol{y})$, the incomplete-data α-log-likelihood ratio $L_{\boldsymbol{y}}^{(\alpha)}(\psi|\varphi)$ should be positive for $\alpha < 1$. As the second term on the right-hand side of (21) is positive before convergence, the positivity of $L_{\boldsymbol{y}}^{(\alpha)}(\psi|\varphi)$ is achieved by maximizing the Q-function $Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(\alpha)}(\psi|\varphi)$.
This continues by replacing φ with ψ until convergence is attained. We note that the case $\alpha = -1$ is the log-EM algorithm. This is verified by using (4) and (13).

Here, we note that (21) can also be derived by directly computing Type-I α-divergence $D_I^{(\alpha)}(\varphi||\psi)$ which measures the divergence from $p(\boldsymbol{x}|\boldsymbol{y},\varphi)$ to $p(\boldsymbol{x}|\boldsymbol{y},\psi)$. Reference [8] adopts this method and gives detailed experimental results on Gaussian mixture distributions.

C. INCOMPLETE-DATA LIKELIHOOD RATIO

Equation (19) can be used to jointly derive the log-EM and α-EM algorithms. Therefore, one may consider the joint maximization of $Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(\alpha)}(\psi|\varphi)$ and $Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\psi|\varphi)$. However, this maximization is more complicated than that of $Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(\alpha)}(\psi|\varphi)$ alone. In addition, the joint incomplete-data likelihood ratio has the following property. Let r stand for $\frac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}$. Then, the function that corresponds to the joint incomplete-data likelihood ratio is

$$w(r) \stackrel{def}{=} \tfrac{2}{1-\alpha}\log r + (\tfrac{2}{1-\alpha})^2(r^{-\frac{1-\alpha}{2}} - 1).$$

$w(r)$ is such that $dw(r)/dr = 0$ at $r = 1$. Therefore, we use the α-EM algorithm.

In the next section, we are concerned with the second objective of our study, that is, the application of the α-EM algorithm as an evaluator for graders. Before this, we highlight the significance of the results that have obtained up to this point.

(a) α-divergence and Bregman divergence are related through probability skewing.
(b) This skewing leads to Type-I and Type-II α-divergence.
(c) Type-II α-divergence is important because it can be used to jointly derive the α-EM and log-EM algorithms.
(d) It is important to note that it is theoretically ensured that the α-EM algorithm has fast convergence. The fast convergence for the exponential family is theoretically ensured by Corollary 8 in [8, p. 699]: There exists β^* such that

$$-1 < \alpha < \beta^* < 1,$$

for which, the spectral radius inequality

$$\rho\left(-\{\partial^{20} Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(\alpha)}(\varphi|\varphi)\}^{-1}\right) > \rho\left(-\{\partial^{20} Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\varphi|\varphi)\}^{-1}\right)$$

holds. When a larger spectral radius is achieved, the convergence of the α-EM algorithm is faster. The right-hand side, that is, the case $\alpha = -1$, corresponds to the log-EM algorithm, which has slow convergence. Here, we use the notation

$$\partial^{ij} f(\varphi|\varphi) \stackrel{def}{=} \frac{\partial^{i+j} f(\psi|\varphi)}{\partial^i \psi\, \partial^j \varphi}\Big|_{\psi=\varphi}.$$

IV. APPLICATION OF α-EM ALGORITHM AS AN EVALUATOR IN A BLOCKCHAIN ENVIRONMENT

Advances in information networks have made it possible for anyone to update content. Consequently, large amounts of raw data may be transferred over a network. However, some of them may be unreliable. Herein, based on the results of the previous sections, we use the α-EM algorithm as an evaluator for data reliability. Here, the data to be evaluated are grader reports. The critical point is that the quality of the evaluators will be subject to further assessment, where network nodes and blockchain environments with various roles appear. Therefore, we begin with an explanation of the network configuration.

A. CREDIT EVALUATION FOR DATA IN BLOCKCHAIN ENVIRONMENT

We first explain the general structure for evaluating data reliability in a blockchain environment. This system is illustrated in Fig. 1. All elements are nodes constituting the internet of collaborative things (IoCT) that can communicate with each other. However, we omit irrelevant edges for simplicity. We explain these nodes starting from the top in the figure.

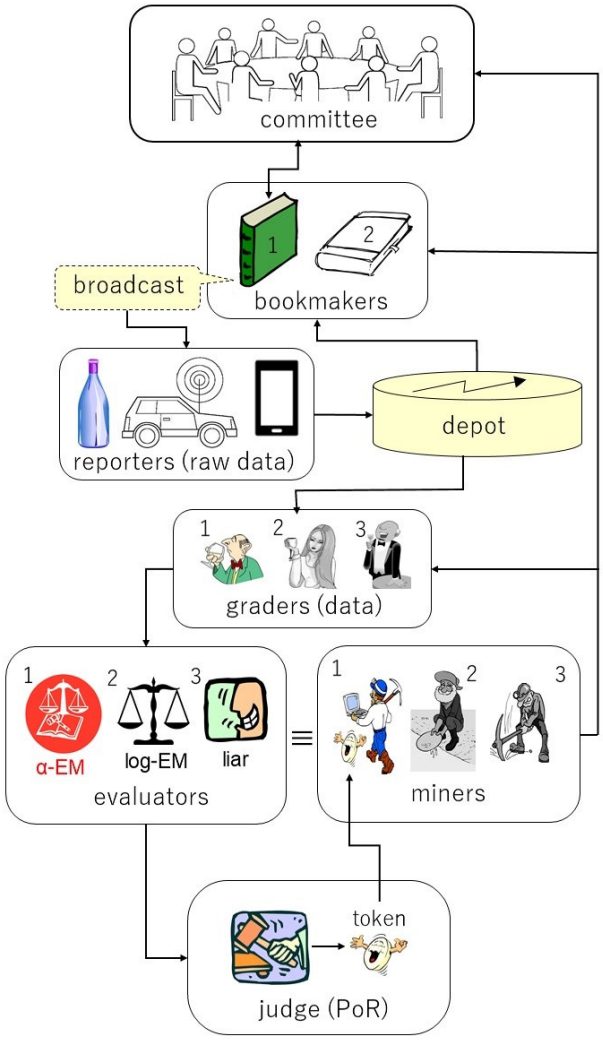

FIGURE 1. Blockchain network by nodes.

However, the evaluators and the judge have the main tasks in this study. Here, evaluation is the qualification of graders by machine learning. It should be recognized that evaluators have different capabilities, and some may be manlike liars. Thus, the central problem is to *further determine the best evaluator*. The best evaluator identifies good graders.

The agents in Fig. 1 are as follows:

Committee: The committee controls the entire data evaluation process: It grants permission to a bookmaker to set a problem, store and pay digital rewards, and ban nonlegal evaluators and miners from carrying out their activities.

Bookmaker: Bookmakers set a problem in which a specified class of data with trustable grading is required. They broadcast this problem and deposits digital rewards (e.g., crypto assets) to the committee. Thus, a bookmaker is a type of dynamic information bank.

Reporter: Reporters send raw data by crowdXing.[2] We regard the submission of objects, such as bottles of wine, as raw data. Reporters receive digital rewards, such as crypto-assets, according to the quality of data.

Depot: If a bookmaker has insufficient storage as a raw data collection point, he/she can borrow a depot by paying a fee.

Grader: Graders provide rating scores on raw data. Graders themselves are further evaluated. Good graders receive digital rewards.

Evaluator: This study is primarily concerned with the evaluation of grading data. Evaluators use machine learning to evaluate the data from graders. Some evaluators may be unreliable or may cheat. Good evaluators identified by the judge receive digital rewards. We compare the α-EM and log-EM algorithms based on the discussion in Section III-B.

Judge: The judge assesses the evaluators on the basis of their qualities. The judge sends a winning token to a selected miner indicating the best evaluator. We place a conscience mechanism there, i.e., a method whereby repeatedly benefiting only specific miners may be avoided. The judge reports the results to the bookmaker and the committee. The judge receives a digital reward from the committee for his/her tasks.

Miner: The miner who received the winning token generates a new block containing the entire data on the bookmaking and digital reward flow. After updating this blockchain as a new ledger, he/she reports its completion to the committee and the bookmaker. The miner also receives a digital reward from the committee.

Before proceeding to the mathematical setting in the next section, we make certain essential comments on the above agents. As shown in Fig. 1, reporters may compile various types of raw data. In the following sections, the raw data are wines so that the terminology becomes straightforward. In this situation, a grader is a sommelier who submits a grading table for wines. The evaluators are *machine learning algorithms* that classify the sommeliers according to the submitted grading table. In this study, the evaluators are machine learning algorithms in the EM-family. Evaluators have strengths and weaknesses: Some of them may act as human liars without reasoning. The judge excludes liars, identifies superior evaluators, and generates a token containing the data on the winning evaluator. The miner who obtains the token can conduct mining. Therefore, we make the following assumptions.

(1) *Evaluators and miners are the same agents*, as shown in Fig. 1. This simplifies the entire system without losing generality. Each evaluator can become a miner by preparing a command execution environment for `miner.start(minerID)`. If the evaluators do not have mining ability, the winning evaluator will pass

[2]CrowdXing stands for crowd{funding, sending, sensing, sourcing, etc.}

the winning token to the appropriate miner. In this case, more space is required in Fig. 1 to describe the communication paths.
(2) For fairness, no evaluator can serve concurrently as the judge.
(3) The committee functions independently of the other agents.
(4) Any agent can be a reporter to send raw data. Henceforth, the raw data will be referred to as wine for clarity.

B. PROBLEM FORMULATION AS SOMMELIER EVALUATION

Let $w \in \boldsymbol{W}$ be a source data point by crowdXing. For each w, each grader $s \in \boldsymbol{S}$ provides a grading score. Before proceeding to the next step, we note the following:

(a) For more clarity, we regard the raw data point w as a wine brand. A grader s is a sommelier. However, this formulation can be applied to a more comprehensive class of trust evaluation problems, as can be understood from Fig. 1.
(b) For notational simplicity, we use the symbol W to denote the cardinality of the raw dataset $\boldsymbol{W}$. Similarly, S denotes the cardinality of the dataset $\boldsymbol{S}$.

To each wine w, the sommelier s assigns a grading score $a \in G$. If s has little confidence in this grading, he/she can report the score multiple times $u_s \in U_s$ for the same wine. However, multiple reports are possible at the cost of a final penalty.

1) INCOMPLETE DATA, MISSING DATA, AND CONFUSION MATRIX

Sommelier $s \in S$ tests wine $w \in W$, u_s times, to report its grading $y^{(s,u_s)}(w)$. The set

$$\boldsymbol{y} = \{y^{(s,u_s)}(w)\}_{w\in W,\ s\in S,\ u_s\in U_s}$$

is the incomplete data to be evaluated. These incomplete data become completed if the set of missing data is obtained:

$$\boldsymbol{z} = \{I_a(w)\}_{w\in W,\ a\in G}.$$

Here,

$$I_a(w) = \begin{cases} 1: \textit{ If the grading } a \textit{ for wine } w \textit{ is true.} \\ 0: \textit{ Otherwise.} \end{cases}$$

Thus, the complete data are $\boldsymbol{x} = \{\boldsymbol{y}, \boldsymbol{z}\}$.

In the formulation of the trust evaluation problem, we use a mixture of multinomial distributions. The notation is as follows:

$n_a^{(s)}(w)$	Number of times that the sommelier s reports the grade a for wine w.
$c_{ab}^{(s)}$	Probability that the sommelier s reports the grade b, whereas grade a is valid.
π_a	Probability that grade a appears by reflecting $n_a^{(s)}(w)$.
$h_a(w)$	Estimation of $I_a(w)$.

Furthermore, the confusion matrix of size $G \times G$ for each sommelier s is

$$C^{(s)} = [c_{ab}^{(s)}]. \tag{22}$$

The confusion matrix is important because sommelier evaluation is based on this quantity.

By using the above definitions, we express the basic probabilities for the estimation of the multinomial mixture distribution. Let the probability of the tasting result $y^{(s,u_s)}(w)$ be

$$P_a(y^{(s,u_s)}(w)) = \prod_{b=1}^{G}\{c_{ab}^{(s)}\}^{n_b^{(s)}(w)}.$$

Here, the unknown true grade of wine w is a. As $c_{ab}^{(s)}$ are probabilities, we have the constraint

$$\sum_{b=1}^{G} c_{ab}^{(s)} = 1.$$

However, there is no constraint on the number of tastings $\sum_{s=1}^{S} n_b^{(s)}(w)$.

2) COMPLETE- AND INCOMPLETE-DATA LIKELIHOOD

For the set of all sommeliers S, we have

$$\boldsymbol{P}_a\left(\prod_{s=1}^{S}\prod_{u=1}^{U_s} y^{(s,u_s)}(w)\right) \stackrel{def}{=} \prod_{s=1}^{S}\prod_{b=1}^{G}\{c_{ab}^{(s)}\}^{n_b^{(s)}(w)} \tag{23}$$

by assuming independence of s and b. Here, multiple tasting u_s is considered in the number of gradings $n_b^{(s)}(w)$.

As the grading is b instead of the unknown true grading a, we use a mixture of multinomial distributions weighted by $\pi_a = P(a)$ such that

$$\sum_{a=1}^{G}\pi_a \boldsymbol{P}_a\left(\prod_{s=1}^{S}\prod_{b=1}^{G} y^{(s,u_s)}(w)\right) = \sum_{a=1}^{G}\pi_a\left\{\prod_{s=1}^{S}\prod_{b=1}^{G}\{c_{ab}^{(s)}\}^{n_b^{(s)}(w)}\right\}.$$

As each data point w is independent of the others, the incomplete-data likelihood in the form of probability becomes

$$P(\boldsymbol{y}) = \prod_{w=1}^{W}\left[\sum_{a=1}^{G}\pi_a\left\{\prod_{s=1}^{S}\prod_{b=1}^{G}\{c_{ab}^{(s)}\}^{n_b^{(s)}(w)}\right\}\right]. \tag{24}$$

The corresponding complete-data likelihood is

$$P(\boldsymbol{x}) = \prod_{w=1}^{W}\prod_{a=1}^{G}\{\pi_a \boldsymbol{P}_a(w)\}^{I_a(w)}. \tag{25}$$

We verify convergence by the incomplete-data likelihood (24). However, its logarithmic form should be used to avoid numerical underflow:

$$\begin{aligned}\mathcal{L}(\boldsymbol{y}) &\stackrel{def}{=} \log P(\boldsymbol{y}) \\ &= \sum_{w=1}^{W}\log\left[\sum_{a=1}^{G}\pi_a\left\{\prod_{s=1}^{S}\prod_{b=1}^{G}\{c_{ab}^{(s)}\}^{n_b^{(s)}(w)}\right\}\right].\end{aligned} \tag{26}$$

3) DERIVATION OF α-EM ALGORITHM FOR THE EVALUATOR

Using the incomplete-data likelihood (24) and complete-data likelihood (25), we can obtain the α-EM algorithm (21) for the evaluation of the sommeliers. The α-log likelihood in (21) is

$$L_{\boldsymbol{y}}^{(\alpha)}(\psi|\varphi) = L_{\boldsymbol{y}}^{(\alpha)}\left(\frac{P(\boldsymbol{y}|\psi)}{P(\boldsymbol{y}|\varphi)}\right). \tag{27}$$

Here, ψ stands for the parameters at the $(\ell+1)$-th iteration, and φ refers to the parameters at the ℓ-th iteration.

The purpose of the α-EM algorithm is to make (27) positive, so that the incomplete-data likelihood continues to increase. This can be achieved by keeping $Q^{(\alpha)}_{\boldsymbol{x}|\boldsymbol{y},\varphi}(\psi|\varphi)$ positive for $\alpha < 1$. Here,

$$Q^{(\alpha)}_{\boldsymbol{x}|\boldsymbol{y},\varphi}(\psi|\varphi) = \tfrac{2}{1+\alpha}\{S^{(\alpha)}_{\boldsymbol{x}|\boldsymbol{y},\varphi}(\psi|\varphi) - 1\}. \tag{28}$$

The S-function is

$$S^{(\alpha)}_{\boldsymbol{x}|\boldsymbol{y},\varphi}(\psi|\varphi) = E_{P(\boldsymbol{x}|\boldsymbol{y},\varphi)}\left[\{\tfrac{P(\boldsymbol{x}|\psi)}{P(\boldsymbol{x}|\varphi)}\}^{\frac{1+\alpha}{2}}\right]. \tag{29}$$

Here, $E_{P(\boldsymbol{x}|\boldsymbol{y},\varphi)}[\cdot]$ stands for the expectation by $P(\boldsymbol{x}|\boldsymbol{y},\varphi)$. Then, the maximization of $Q^{(\alpha)}_{\boldsymbol{x}|\boldsymbol{y},\varphi}(\psi|\varphi)$ is equivalent to that of $S^{(\alpha)}_{\boldsymbol{x}|\boldsymbol{y},\varphi}(\psi|\varphi)$.

As we have the expression for the complete-data probability (25), we express the S-function using the indicators for the missing data as follows. Hereafter, we denote the true grade by g because we use the indices a and b for dummy variables.

$$\begin{aligned} &S^{(\alpha)}_{\boldsymbol{z}|\boldsymbol{y},\varphi}(\psi|\varphi) \\ &= E_{P(\boldsymbol{z}|\boldsymbol{y},\varphi)}\left[\left\{\frac{\prod_{w=1}^{W}\prod_{g=1}^{G}\{\pi_g^{\psi}\boldsymbol{P}_g^{\psi}(w)\}^{I_g(w)}}{\prod_{w=1}^{W}\prod_{g=1}^{G}\{\pi_g^{\varphi}\boldsymbol{P}_g^{\varphi}(w)\}^{I_g(w)}}\right\}^{\frac{1+\alpha}{2}}\right] \\ &= \prod_{w=1}^{W}\sum_{g=1}^{G} h_g(w)\left\{\frac{\pi_g^{\psi}\boldsymbol{P}_g^{\psi}(w)}{\pi_g^{\varphi}\boldsymbol{P}_g^{\varphi}(w)}\right\}^{\frac{1+\alpha}{2}}. \end{aligned} \tag{30}$$

Here,

$$\begin{aligned} h_g(w) &= P\left(I_g(w)=1 \,\middle|\, \prod_{s=1}^{S} y^{(s,u_s)}(w), \varphi\right) \\ &= \frac{\pi_g \prod_{s=1}^{S}\prod_{b=1}^{G}\{c^{(s)}_{gb}(\varphi)\}^{n_b^{(s)}(w)}}{\sum_{a=1}^{G}\pi_a \prod_{s=1}^{S}\prod_{b=1}^{G}\{c^{(s)}_{ab}(\varphi)\}^{n_b^{(s)}(w)}}, \end{aligned} \tag{31}$$

where $c^{(s)}_{ab}(\varphi)$ denotes $c^{(s)}_{ab}$ under the probability parameter of φ. The second equality in (31) follows from Bayes' theorem. In this equation, we denote the right-hand side by $h_g(w)$, suppressing the symbol φ for notational simplicity. Computing (31) is the E-step.

For the α-EM algorithm, we should compute $\boldsymbol{P}_g^{\psi}(w)$ and $\boldsymbol{P}_g^{\varphi}(w)$ in (30). This implies that $\boldsymbol{P}_a(w)$ in (23) is computed using $c^{(s)}_{ab}(\psi)$ and $c^{(s)}_{ab}(\varphi)$, and then a is replaced by g. We provide the details of this derivation in Appendix C.

Then, we proceed to the following:

$$\begin{aligned} S^{(\alpha)}_{\boldsymbol{z}|\boldsymbol{y},\varphi}(\psi|\varphi) &= \prod_{w=1}^{W}\left\{\sum_{g=1}^{G} h_g^{(\alpha)}(w)\right\} \\ &\stackrel{def}{=} \prod_{w=1}^{W} V^{(\alpha)}(w). \end{aligned} \tag{32}$$

Here,

$$h_g^{(\alpha)}(w) = h_g(w)\left\{\frac{\pi_g^{\psi}\boldsymbol{P}_g^{\psi}(w)}{\pi_g^{\varphi}\boldsymbol{P}_g^{\varphi}(w)}\right\}^{\frac{1+\alpha}{2}}. \tag{33}$$

We normalize $h_g^{(\alpha)}(w)$ to obtain

$$\tilde{h}_g^{(\alpha)}(w) = \frac{h_g^{(\alpha)}(w)}{V^{(\alpha)}(w)} \tag{34}$$

so that

$$\sum_{g=1}^{G}\tilde{h}_g^{(\alpha)}(w) = 1. \tag{35}$$

Therefore, the E-step of the α-EM algorithm comprises two steps: The first is the computation of (31), which is the same as the E-step of log-EM. The second is the computation of (34), which is termed the A-step or acceleration step.

For the M-step, we apply direct differentiations including Lagrange multipliers. In the update of π_g^{ψ}, we have

$$\pi_g^{\psi} = \frac{\sum_{w=1}^{W}\tilde{h}_g^{(\alpha)}(w)}{W}. \tag{36}$$

For $c^{(s)}_{gb}(\psi)$, we have

$$c^{(s)}_{gb}(\psi) = \frac{\sum_{w=1}^{W}\tilde{h}_g^{(\alpha)}(w)n_b^{(s)}(w)}{\sum_{w=1}^{W}\sum_{a=1}^{G}\tilde{h}_g^{(\alpha)}(w)n_a^{(s)}(w)}. \tag{37}$$

We provide more details regarding the derivation of the M-step, in addition to the E-step and A-step, in Appendix C.

Thus, we have the following α-EM steps.

[Update iterations for the α-EM algorithm]

E-step: Compute (31).
A-step: Compute (34). This step is omitted from the first cycle, where we use $h_g(w)$ instead of $\tilde{h}_g^{(\alpha)}(w)$.
M-step1:Compute (36).
M-step2:Compute (37).
C-step: If convergence is achieved, then stop; otherwise, go back to E-step.

Convergence is analyzed using the incomplete-data log-likelihood (26). In addition to the log-likelihood, we use the maximum ripple computed on the probabilities of $h_g(w)$, π_g^{ψ}, and $c^{(\alpha)}_{gb}(\psi)$.

4) FACTORS OF SOMMELIER RANKING

The evaluator uses the confusion matrix (22), which is computed by (37), to evaluate the sommeliers. The evaluator computes the following $Q(s)$ for each sommelier s:

$$\mathcal{Q}(s) = \sum_{g=1}^{G}\left[f_1(g)\left\{\sum_{b=1}^{G} f_2(g,b)c^{(s)}_{gb}\right\}\right]. \tag{38}$$

Here, $f_1(g)$ is either 1 or π_g, and $f_2(g,b)$ is either 1 or a weight that reflects the distance from the diagonal position. The case $f_1(g) = 1$ and $f_2(g,b) = \delta_{gb}$ is the trace of the confusion matrix $C^{(s)}$ in (22). If a sommelier s tried multiple grading, $\mathcal{Q}(s)$ is multiplied by a penalty factor.

V. VERIFICATION OF α-EM ALGORITHM BY NUMERICAL DATA

A. PREPARATION OF RAW DATA AND GRADING TABLE

As discussed in the previous section, we use wines as raw data. The Kaggle site [20] provides a large number of wine brands from around the world and their reputation. However, the following preprocessing steps are required.

(1) Removing textual comments.
(2) Rescaling the grades from 0 to 10. This corresponds to the grades from 80 to 100 on the Parker scale. The range from 0 to 10 (rather than from 0 to 20) is used owing to memory limitations of conventional PC.
(3) As there are many wine brands on the Kaggle site, random sampling is necessary. The triangular distribution

of the source data should be maintained. Rare wine brands of grade 10 should be included as long as the triangular distribution is maintained.

By using the above conditions, we generated a grading table, given the memory limitations of the PC. The PC used in the experiments has a main memory of 8 GB and runs at 1.80×2 (effective speed: 2.39) GHz. Accordingly, we generated a table with a size of $W \times S = 750 \times 25 = 18,750$. We note that the scalability of the α-EM algorithm on the mixture of multinomial distributions is only due to the main memory size.

B. ADAPTATION OF α-EM ALGORITHM TO DISCRETE DATA

When discrete data are handled, a zero-division check for the numerical exception is necessary.

(a) If $\boldsymbol{P}_g^{\psi}(w) = \boldsymbol{P}_g^{\varphi}(w) = 0$, we update (33) by
$$h_g^{(\alpha)}(w) = h_g(w)(\pi_g^{\psi}/\pi_g^{\varphi})^{(1+\alpha)/2}.$$
(b) If $\boldsymbol{P}_g^{\psi}(w) > 0$ and $\boldsymbol{P}_g^{\varphi}(w) = 0$, we use $h_g^{(\alpha)}(w) = h_g(w)$ for the update. This method corresponds to adopting $\alpha = -1$.
(c) In case of ill-conditioned tables with $\pi_g = 0$, grade g should be removed in advance. However, this is not the case with the prepared data table.

C. SELECTING AND CONTROLLING α

The region $\alpha < 1$ is the theoretically possible region. The case $\alpha = -1$ is the traditional log-EM algorithm. In the discussion on the spectral radius, it was pointed out that determining $\beta^* \in (-1, 1)$ is important to achieve fast convergence of the α-EM algorithm (see III-C(d)). As β^* depends on empirical data, selecting and controlling the parameter α is essential. This situation is common in any optimization problem applied to empirical data. Among the four methods provided, we recommend the dynamic tuning method.

1) CONSTANT α

This method, in which an α larger than -1 is selected, is the simplest; however, it often fails. A large value of α may cause go-and-back in the iteration of the algorithm, which is common in any optimization problem. Then, the choice $\alpha = 0$, which corresponds to Hellinger-distance optimization, becomes a plausible method. However, experiments demonstrate that the speed gain is not as high as expected.

2) GREEDY SHOTGUN OR FULL SHOTGUN FOR PARALLEL α

Inspired by the constant-α method, we use parallelism on the A-step and M-step in (34), (36), and (37). We prepare a set of α such that $\mathcal{A} = \{\alpha_0, \alpha_1, \ldots, \alpha_N\}$ in ascending order. Here, $\alpha_0 = -1$. After completing the E-step in (31), we can compute the {A, M}-steps in parallel. Then, the algorithm selects an α_n that yields the largest log-likelihood $\mathcal{L}(\boldsymbol{y})$. If the log-likelihood $\mathcal{L}(\boldsymbol{y})$ becomes smaller than in the previous cycle for any α_n in $\mathcal{A}$, we select $\alpha_0 = -1$. We term this method greedy shotgun or full shotgun. In greedy shotgun, the choice of α_N is important. Regarding the size N, we cannot provide a processor for each α_n because that system is quite expensive. Therefore, we select $\alpha_N < 1$ for the level of the multi-core PC. If we insist on $\alpha_N = 1$, then the speed gain diminishes. If this choice is not properly made, the greedy shotgun may be problematic. This is investigated in the experiments.

3) ORDERED SHOTGUN METHOD

The ordered shotgun method also uses the set $\mathcal{A} = \{\alpha_0, \alpha_1, \ldots, \alpha_N\}$, which is in ascending order. In this case, $\alpha_0 = -1$ and $\alpha_N \leq 1$. In this method, we begin with $\alpha = \alpha_N$. If the log-likelihood $\mathcal{L}(\boldsymbol{y})$ from the previous cycle increases, we accept this α; otherwise, we decrease α to α_{N-1}, and so on. In this method, we can observe the following:

(a) Unlike in the case of the greedy shotgun, the selected α may not maximize $\mathcal{L}(\boldsymbol{y})$ best among $\mathcal{A}$; however, it increases the likelihood.
(b) The necessary computational resources are far less than for the greedy shotgun method because this method is sequential.
(c) We can begin with $\alpha_N = 1$. However, an appropriately selected $\alpha_N < 1$ leads to faster convergence because changes to α become less frequent.

4) DYNAMIC TUNING

As the log-likelihood $\mathcal{L}(\boldsymbol{y})$ is checked at each iteration, it is possible to tune the parameter α dynamically. If $\mathcal{L}(\boldsymbol{y})$ is decreased by the previous iteration by a large α, it becomes necessary to reduce its value. We perform this dynamic tuning of α as follows: We first select $\alpha_{\max}$ and $\alpha_{\min}$. If $\mathcal{L}(\boldsymbol{y})$ is increased or retained, we accept this cycle and increase α by
$$\alpha := \alpha + \eta(\alpha_{\max} - \alpha) \tag{39}$$
for the next iteration. Here, we select $\eta \in (0, 1)$ as a pre-specified design parameter. If $\mathcal{L}(\boldsymbol{y})$ is decreased, we recompute the A-step of this cycle using $\alpha_{\min}$. The experiments demonstrate that using $\alpha_{\max} = 1$, $\alpha_{\min} = -1$, and $\eta = 0.1$ leads to a remarkable speed increase.

D. EXPERIMENTS ON SOMMELIER EVALUATION

The log and α-EM algorithms were used as evaluators for the prepared dataset. Although the log-EM algorithm is a special case of $\alpha = -1$, we considered this case independently. Therefore, we can fairly compare the performance of the EM-family {log, constant α, greedy shotgun, ordered shotgun, dynamic tuning}.

1) PRACTICAL SETTINGS FOR EXPERIMENTS

In the experiments, we set standard rules for a fair comparison.

(a) Regarding the initial value for the 0-th E-step, we use
$$h_g(w) = \frac{\sum_{s=1}^{S} n_g^{(s)}(w)}{\sum_{b=1}^{G} \sum_{s=1}^{(s)} n_b^{(s)}(w)}. \tag{40}$$

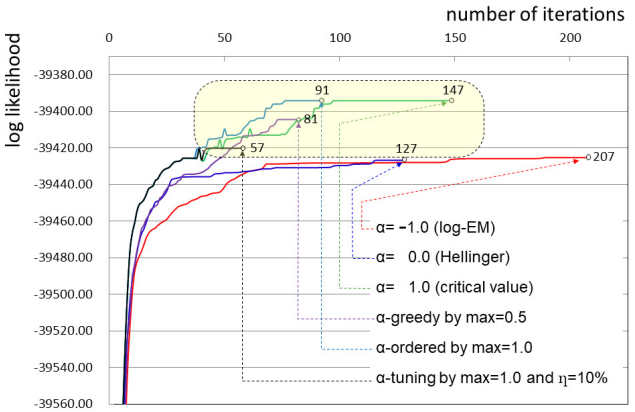

FIGURE 2. Comparison of convergence speed.

(b) We set the numeric convergence criterion as follows: We first select a window size of L for the iterations; for example, $L = 3$. Then, we verify whether both of the following criteria are satisfied:

(1) We determine whether the following inequality holds for iterations $\ell > L$:

$$\mathcal{L}_{avg} = \left| \frac{\mathcal{L}_\ell - \mathcal{L}_{\ell-L}}{L(\mathcal{L}_\ell - \mathcal{L}_L)} \right| < \varepsilon_{\mathcal{L}} \tag{41}$$

Here, $\varepsilon_{\mathcal{L}}$ is a prespecified small positive number.

(2) We apply a similar criterion for the convergence of probabilities. We verify whether the ripples of the probabilities π_a, $h_a(w)$, and $c_{ab}^{(s)}$ are sufficiently small. Let p_ℓ stand for these probabilities at cycle ℓ. Then, we determine whether the following inequality holds:

$$\max |p_\ell / p_{\ell-1}| < \varepsilon_{\mathcal{P}} \tag{42}$$

Here, $\varepsilon_{\mathcal{P}}$ is a prespecified small positive number.

2) EXPERIMENTS ON NUMBER OF ITERATIONS

After these preparations, we conducted evaluator comparison experiments. Fig. 2 shows the convergence speed in terms of the number of iterations. The horizontal axis is the scale for the number of iterations. The vertical axis corresponds to the incomplete-data log-likelihood $\mathcal{L}(\boldsymbol{y})$ in (26). As it is the logarithm of the probability, the axis values are negative. Larger values imply better optima.

There are six curves, or six evaluators. At the end of each curve, we indicate the required number of iterations as a numeral. There are three types of constant α and three types of adjusting α. In the constant case, we consider the cases $\alpha = \{-1, 0, 1\}$. The log-EM corresponds to $\alpha = -1$. However, we considered this case independently of the α-EMs because it provides the baseline for every comparison. In the case of varying α, we consider the methods of {greedy, ordered}-shotguns and dynamic tuning.

From the experimental results shown in Fig. 2, we observe the following regarding the required number of iterations and converged log-likelihood. We note that these results were obtained with additional experiments that are not shown in the figure.

Log-EM (207 cycles): Convergence is slow even for the discrete-value evaluation problem. Although the common initial value in (40) is reasonable, the converged log-likelihood is inferior to that of the α-EMs.

Constant $\alpha = 0$ (127 cycles): We experimented with the case $\alpha = 0$ because it corresponds to Hellinger-distance optimization. This distance measures the square of the difference between two square-root probabilities. This case involves a smaller number of iterations than log-EM. The converged log-likelihood is close to that of log-EM. A few decreases in the log-likelihood occurred during the iterations.

Constant $\alpha = 1$ (147 cycles): As the constant $\alpha = 1$ is the critical value for convergence, we use this result only for reference. The converged log-likelihood is better than that of log-EM and $\alpha = 0$. However, the number of required cycles is greater than for $\alpha = 0$. This is because frequent decreases in the log-likelihood caused re-computations by $\alpha = -1$, thus adding cycles. The findings of this experiment led to the following methods for α-adjusting.

Greedy shotgun (81 cycles): This method is the first step for improving the defects of the constant $\alpha = 1$. As explained in Section V-C2, we prepared the parameter set $\mathcal{A}$ with equally spaced elements of $N = 7$. We always selected $\alpha_0 = -1$, that is, the case log-EM. In the initial experiments, we selected a larger α_N close to 1. In these cases, log-likelihoods inferior to those of log-EM were observed. Therefore, we selected $\alpha_N = 0.5$ so that $-1 \le \alpha \le 0.5$. This case outperformed log-EM by far in terms of both iteration cycles and the converged log-likelihood. However, the actual advantage is seen if N parallel processors with little communication delay are provided.

Ordered shotgun (91 cycles): This method is an improvement of the greedy shotgun so that $\alpha_N = 1$ is allowed. If the choice of α_N yields a log-likelihood decrease, the parameter becomes α_{N-1}, and so on. The number of iterations is slightly higher than that of the greedy shotgun; however, it is better than that of log-EM by far. The converged log-likelihood is close to the value for the greedy shotgun, which is better than for log-EM.

Dynamic tuning (57 cycles): An improvement of the ordered shotgun is dynamic tuning. This method relaxes the preselection of $\mathcal{A}$ in the ordered shotgun method. In the tuning method, the set $\mathcal{A}$ becomes the entire region $[-1, 1]$ without requiring parallel processors. The only preselection is on the recovery rate of $\eta = 0.1$ for α. Convergence speed is the highest among the methods in Fig. 2. The

converged log-likelihood is inferior to that of the ordered shotgun method but better than that of log-EM. The round-corner rectangle region shown in Fig. 2 indicates the effective area that gives the same opinion on the sommeliers. We explain the evaluation method related to the blockchain environment below. The pseudo code for dynamic tuning in C-language style is provided in Appendix D.

E. COMPARISON OF CPU TIME

If the computational complexity of the α-EM algorithms far exceeds that of log-EM, the speed gain in terms of the number of iterations shown in Fig. 2 will be lost. However, this is not an issue even for serial computing PCs. The following experiments support the advantages of the ordered shotgun and dynamic tuning methods.

We note that the number of required iterations remains the same, as shown in Fig. 2. However, actual run time varies even for fixed programming code. This fluctuation is due to the processing speed of the program depending on the OS status. Therefore, we used an average time of 10 runs for each method, and observed the following.

Log-EM: The average runtime was 8601.8 ms for 207 iterations. Therefore, the average run time for one cycle was 8601.8/207=41.55 ms. We used this value as a baseline for comparison with the α-EM algorithms.

Constant $\alpha = 0$:This case required 5400.9 ms for 127 iterations. The speed gain as measured by runtime over log-EM was 1.593. One cycle required 42.53 ms.

Constant $\alpha = 1$:This case required 6668.4 ms for 147 iterations. The speed gain as measured by runtime over log-EM was 1.290. One cycle required 45.36 ms. Thus, the one-cycle overhead to log-EM was approximately 6% because $(42.53 + 45.36)/2/41.55 = 1.058$. Therefore, the overhead determined using the A-step was negligible compared with the speed gain of α-EM.

Greedy shotgun: This method implemented by a serial processor exhibited no benefits. The speedup ratio was 0.642, which is inferior to that of log-EM. However, CPU time by simulated parallel processing exhibited a speed increase ratio of 3.353 over log-EM.

Ordered shotgun: This method with $\alpha_N = 1$ converged after 91 iterations, which required 7532.0 ms. One cycle required 80.79 ms, which is approximately twice as much as the time required by log-EM. The reason is that the decrease from α_n to α_{n-1} occurs in several cycles. However, the speed increase ratio was $8601.8/7352.0 = 1.170$ over log-EM. By selecting $\alpha_N = 0.5$, convergence was attained after 74 iterations, which required 3406.3 ms. Then, the speed increase ratio was $8601.8/3406.3 = 2.525$. This value is close to the upper bound $207/74 = 2.797$, which is the speed increase ratio in terms of iteration count. This observation led to the dynamic tuning method.

Dynamic tuning: This method required 2574.0 ms until convergence after 57 iterations. Therefore, the speed increase ratio over log-EM was $8601.8/2574.0 = 3.342$. This number is close to $207/57 = 3.632$, which is the upper bound of the speed increase in terms of the iteration number in Fig. 2. In addition, the one-cycle overhead to log-EM was approximately 9%, arising from the ratio $45.16/41.55 = 1.087$. This overhead was negligible compared with the speed gain.

F. PREPARATION FOR ASSESSMENT

For each evaluator, there are two more steps after the convergence of the algorithm.

(1) Each evaluator, for example e, prepares reporting data as follows:

$$\mathcal{D}^{(e)} = \Big\{ \mathcal{L}(\mathbf{y}), \{\pi_a\}_{a=1}^{G}, \{c_{ab}^{(s)}\}_{a=1\ b=1}^{G\ G}, \{n_b^{(s)}(w)\}_{b=1\ w=1}^{G\ W}, \{t_{start},\ t_{end},\ f_{th}\} \Big\}^{(e)}.$$

Here, t_{start} and t_{start} are timestamps [21] that indicate the algorithm execution. f_{th} is the theoretical FLOPS of the computer used by the evaluator:

$$f_{th} = (CPU\ clock\ freqency) \times (\#\ of\ operations/clock) \times (\#\ of\ CPUs).$$

The evaluator e performed

$$t_e = (t_{end} - t_{start}) \times f_{th}$$

theoretical floating-point operations.

(2) The evaluator e generates a pair of a public key $\mathcal{K}_{pub}^{(e)}$ and its secret key $\mathcal{K}_{sec}^{(e)}$. Then, by using a hash function $H(\cdot)$ and the secret key $\mathcal{K}_{sec}^{(e)}(\cdot)$, the evaluator e generates an encrypted hash value[3]:

$$\mathcal{H}^{(e)} = \mathcal{K}_{sec}^{(e)}(H(\mathcal{D}^{(e)})).$$

The evaluator e sends the pair of data $(\mathcal{D}^{(e)}, \mathcal{H}^{(e)})$ to the judge, as shown in Fig. 1.

VI. PROOF-OF-REVIEW FOR MINER ASSIGNMENT IN BLOCKCHAIN BY JUDGING EVALUATOR COMPETENCE

As shown in Fig. 1, the judge is the last part of the selection process for the champion evaluator. As each evaluator has mining ability, the main task of the judge is to identify the best evaluator, who is also the miner. We use the PoR strategy, which is a type of PoS.

A. SCRUTINY OF TRANSMITTED DATA

The judge makes two levels of checks to the data sent from evaluators.

[3]The evaluator e can use the Linux command `openssl` to generate a pair of cryptographic keys $\{\mathcal{K}_{pub}^{(e)}, \mathcal{K}_{sec}^{(e)}\}$, and to hash by `sha256sum` and `rmd160`.

1) COUNTERMEASURES AGAINST SPOOFING

In a network environment, an impersonation check is necessary. We assume that the judge received the dataset $(\mathcal{D}^{(e)}, \mathcal{H}^{(x)})$ from person x who claims to be evaluator e. Then, the judge performs Diffie–Hellman authentication [22] using the public key of the evaluator. The judge computes

$$\tilde{T} = \mathcal{K}_{pub}^{(e)}(\mathcal{H}^{(x)}).$$

If $\tilde{T} = H(\mathcal{D}^{(e)})$, the nominal evaluator x is the authentic evaluator e. If this equality does not hold, the data-sender impersonates the evaluator.

2) INSPECTION OF FORGED DATA

Authenticated evaluators could have sent forged data to the judge. The judge can detect counterfeit data as follows.

(1) Through correctness of convergence: The judge recomputes the incomplete-data log-likelihood (26). If the result matches the sent data, the data are correct; otherwise, the data are fabricated. We know that realizing a good log-likelihood value under floating-point computation is machine learning itself.
(2) Through correctness of computing time (t_{start} and t_{end}): The timestamp [21] becomes a certificate.

B. IDENTIFICATION OF WINNING EVALUATOR

To explain the best evaluator identification method, a set of evaluators should be examined by the judge. We have a group of evaluators with six datasets obtained by the experiments in Fig. 2.

1) ASSESSMENT FACTORS

Let $\mathcal{E}_{passed} = \{E_1, \cdots, E_e, \cdots, E_M\}$ be the evaluators who passed the authentication and forging check of Section VI-A2. The assessment criterion is a combination of the achieved log-likelihood and computing speed. It consists of five steps:

Step 1: Compute the normalized log-likelihoods:

$$\overline{\mathcal{L}(\mathbf{y})}_e = \frac{\max_j \mathcal{L}(\mathbf{y})_j - \mathcal{L}(\mathbf{y})_e}{\max_j \mathcal{L}(\mathbf{y})_j - \min_j \mathcal{L}(\mathbf{y})_e} \in [0, 1].$$

Here, e stands for E_e for notational simplicity, and the same is true for j. A smaller value implies better log-likelihood.

Step 2: Compute the normalized operation count, which is a measure of the relative operation count:

$$\bar{t}_e = \frac{t_e - \min_j t_j}{\max_j t_j - \min_j t_j} \in [0, 1].$$

A smaller value implies a better score.

Step 3: Using a prespecified β, the judge computes the following combination:

$$r_e = \beta_e \overline{\mathcal{L}(\mathbf{y})}_e + (1 - \beta)\,\bar{t}_e. \tag{43}$$

Step 4: The judge imposes a handicap using a conscience mechanism [23] if there are sufficiently many past applications:

$$q_e = \frac{(\#\ of\ past\ wins\ by\ e) + 1}{(\#\ of\ past\ entries\ by\ e) + 1},$$
$$h_e = q_e / \Sigma_j q_j. \tag{44}$$

Step 5: The judge selects the winning evaluator by

$$w = \arg\min_{e \in \mathcal{E}_{passed}} (h_e r_e). \tag{45}$$

2) SOMMELIER SPECIFIED BY WINNING EVALUATOR

As was specified by (38), each sommelier s (a grader in Fig. 1) provides the confusion matrix (22). Only the winning evaluator can provide the ordering of the sommeliers using these confusion matrices $\{C^{(s)}\}_{s=1}^{S}$. The diagonal elements of these matrices indicate correct estimation for each grade g. For each g, the number of wines differs. Therefore, we weight each element of the confusion matrix by $f_1(g) = \pi_g$. Regarding the selection of $f_2(g, b)$, we examine two cases. The first is $f_2(g, b) = \delta_{gb}$, in which only correct answers are credited. The second is to provide a weight that becomes smaller as the distance from the correct answer increases. Through these experiments, we arrive at the following conclusions:

(1) The first method, which counts only the correct answers, is better than the second. By using the second method, the weighting $f_2(g, b)$ indicates the tolerance of the examiner and the arbitrariness of the champion. Therefore, we select the first method: $f_1(g) = \pi_g$ and $f_2(g, b) = \delta_{gb}$.
(2) In the experiments shown in Fig. 2, log-EM failed to identify the champion and runner-up sommeliers who were selected by the α-EM methods. This is observed when both the first and the second method described above (1) are used. Therefore, in Fig. 2, we specified a round-corner rectangle region. We accept only the sommelier ordering reported by good evaluators in that region.

C. ISSUANCE OF WINNING TOKEN

The final task for the judge is to issue a winning token to the best evaluator based on (45). This method is PoR, which is a type PoS. The winning token comprises the following information:

$$\mathcal{I} = \{\text{CommitteeID, BookmakerID, ProblemID,}$$
$$\text{JudgeID, BestEvaluatorID,}$$
$$\text{BestGraderID and its Grading Table}\}.$$

Communication related to the winning token is subject to the Diffie–Hellman authentication method, as explained in Sections V-F and VI-A.

D. MINING AND SUBSEQUENT INSPECTIONS

An eligible miner is the winning evaluator who receives the winning token from the judge. This miner adds a block of data $\mathcal{I}$ as a ledger to the blockchain. Then, we reach the final stage of the entire procedure.

Report: Upon receiving the completion of the blockchain operation, the judge reports the successful evaluation to the committee and bookmaker.

Payment: The bookmaker pays digital rewards to {committee, depot, best-grader, best-evaluator, judge, miner, reporters}, using the funds deposited to the committee.

Normally, this stage is the end of the procedure.

Nothing-at-Stake attack:There could exist an unpaid malicious miner who attempts to add a wrong ledger. If this occurs, the committee adds a block indicating that the wrong ledger is invalid. The committee deprives this miner of mining rights and confiscates any deposits.

E. FINDINGS FROM ASSESSMENT EXPERIMENTS

1) ASSESSMENT OF THE BEST EVALUATOR

We use $\beta = 0.5$ in (43), which considers the normalized log-likelihood and the normalized operation count equally. We use $h_e \equiv 1$ in (44) because of the first bookmaking. Then, the tuning α-EM method is the best evaluator. It reported that the sommelier with ID-number 8 is the most reliable, who is the sommelier that is recognized by the majority of the evaluators. Exceptions are constant-α with $\alpha = 0$ and log-EM, which reported that the sommelier with ID-number 4 would be the best. However, log-EM is the worst evaluator; it lost the competition because of the slow convergence to an inferior optimum.

2) ONE-WAY PROPERTY

In Section VI-A2, illegal data were detected by using the compatibility of $\mathcal{L}(\boldsymbol{y})$ with $\{\pi_a\}_{a=1}^G$ and $\{C^{(s)}\}_{s=1}^S$. We now construct counterfeit data that would pass this compatibility test. This process comprises two steps:

Step 1: Select a desirable numerical value for $\mathcal{L}(\boldsymbol{y})$ that would be the champion recognized by the judge.

Step 2: Determine $\{\pi_a\}_{a=1}^G$ and $\{C^{(s)}\}_{s=1}^S$ that generate $\mathcal{L}(\boldsymbol{y})$ using (26).

If the above steps were possible, the set $\{\mathcal{L}(\boldsymbol{y}), \{\pi_a\}_{a=1}^G, \{C^{(s)}\}_{s=1}^S\}$ reported to the judge could pass the check for data forging. The reverse authentication process from the above Step 2 to Step 1 using (26) is easy. However, there exists a one-way property: The computation in Step 2 after the constraint in Step 1 requires an excessively large amount of computational power. This process requires the floating-point computation of the order of "2 to the double-precision bits." If someone could succeed in this task, the judge would accept the result as valid. However, the estimation method for $\{\{\pi_a\}_{a=1}^G, \{C^{(s)}\}_{s=1}^S\}$ in this study is significantly faster and can overcome the aforementioned method.

VII. CONCLUSION

We investigated the interrelationships among three topics. The first is the compatibility of divergence families by skewing probability spaces, presenting a new α-divergence. This α-divergence led to the joint derivation of the log-EM and α-EM algorithms. The second is the utilization of the α-EM algorithm for further evaluation of graders by reviewing their grading tables. We modeled this problem using a mixture of multinomial distributions. The α-EM algorithm clearly outperformed the log-EM algorithm. The third is the use of the α-EM algorithm for consensus-building in mining. This method is termed PoR because of the computational review process for the judgment of the evaluators.

We focused on sommelier evaluation to avoid lengthy terminology. However, we can apply this trust measurement formulation to various types of counting data exchanged over a network. In this sense, we realized a dynamic information bank.

APPENDIX A
DERIVATION OF THE RELATIONSHIP BETWEEN TYPE-I AND TYPE-II α-DIVERGENCES

We start from (14) of the case $\alpha \neq 1$.

$$\begin{aligned} D_{II}^{(\alpha)}(p||q) &= (\tfrac{2}{1-\alpha})^2 \int_{\mathcal{X}} p\{(\tfrac{q}{p})^{\frac{1-\alpha}{2}} - \log(\tfrac{q}{p})^{\frac{1-\alpha}{2}} - 1\}d\boldsymbol{x} \\ &= (\tfrac{2}{1-\alpha})^2 \int_{\mathcal{X}} p\{(\tfrac{q}{p})^{\frac{1-\alpha}{2}} - 1\}d\boldsymbol{x} \\ &\quad + \tfrac{2}{1-\alpha} \int_{\mathcal{X}} p \log(\tfrac{p}{q})d\boldsymbol{x} \\ &= \tfrac{2}{1-\alpha} \int_{\mathcal{X}} p \log(\tfrac{p}{q})d\boldsymbol{x} \\ &\quad - \tfrac{1+\alpha}{1-\alpha}\tfrac{4}{1-\alpha^2}\{1 - \int_{\mathcal{X}} p(\tfrac{q}{p})^{\frac{1-\alpha}{2}}\}d\boldsymbol{x} \\ &= \tfrac{2}{1-\alpha}K(p||q) - \tfrac{1+\alpha}{1-\alpha}D_I^{(-\alpha)}(p||q). \end{aligned}$$

The second term vanishes through the substitution $\alpha = -1$. The case $\alpha = +1$ follows by using L'Hôspital's rule.

APPENDIX B
RELATIONSHIPS AMONG TYPE-II α-DIVERGENCE AND LIKELIHOOD RATIOS

We first compute $K(p(\boldsymbol{x}|\boldsymbol{y}, \varphi)||p(\boldsymbol{x}|\boldsymbol{y}, \psi)) \stackrel{def}{=} K(\varphi||\psi)$, which is the Kullback–Leibler divergence of (9). For the case $\alpha \neq 1$, we have

$$\begin{aligned} &\tfrac{2}{1-\alpha}K(\varphi||\psi) \\ &\quad = \tfrac{2}{1-\alpha}\log\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)} - \tfrac{2}{1-\alpha}\int_{\mathcal{X}(\boldsymbol{y})} p(\boldsymbol{x}|\boldsymbol{y}, \varphi)\log\tfrac{p(\boldsymbol{x}|\psi)}{p(\boldsymbol{x}|\varphi)}d\boldsymbol{x} \end{aligned} \tag{B.1}$$

because of (15). Similarly, we have the following equality:

$$\begin{aligned} &\tfrac{1+\alpha}{1-\alpha}D_I^{(-\alpha)}(\varphi||\psi) \\ &= (\tfrac{2}{1-\alpha})^2[1 - \{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\}^{-\frac{1-\alpha}{2}}] \\ &\quad - (\tfrac{2}{1-\alpha})^2\{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi)}\}^{-\frac{1-\alpha}{2}}\int_{\mathcal{X}(\boldsymbol{y})} p(\boldsymbol{x}|\boldsymbol{y}, \varphi)[\{\tfrac{p(\boldsymbol{x}|\psi)}{p(\boldsymbol{x}|\varphi)}\}^{\frac{1-\alpha}{2}} - 1]d\boldsymbol{x}. \end{aligned} \tag{B.2}$$

Then, by subtracting (B.2) from (B.1), we have

$$\begin{aligned} &D_{II}^{(\alpha)}(\varphi||\psi) \\ &\quad = \tfrac{2}{1-\alpha}L_{\boldsymbol{y}}^{(-1)}(\psi|\varphi) - \tfrac{2}{1-\alpha}\{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi}\}^{-\frac{1-\alpha}{2}}L_{\boldsymbol{y}}^{(-\alpha)}(\psi|\varphi) \\ &\quad - \tfrac{2}{1-\alpha}Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-1)}(\psi|\varphi) + \tfrac{2}{1-\alpha}\{\tfrac{p(\boldsymbol{y}|\psi)}{p(\boldsymbol{y}|\varphi}\}^{-\frac{1-\alpha}{2}}Q_{\boldsymbol{x}|\boldsymbol{y},\varphi}^{(-\alpha)}(\psi|\varphi) \end{aligned}$$

by (14).

APPENDIX C
DERIVATION OF E-STEP, A-STEP, AND M-STEP

A. DERIVATION OF E-STEP AND A-STEP

The derivation of E-step or (31) follows [11]. By using Bayes' theorem applied to the first line of (31), we have

$$h_g(w) = P_1 P_2 / P_3.$$

Here,

$$\begin{aligned} P_1 &= P\left(\prod_{s=1}^{S}\prod_{u=1}^{U_s} y^u(w) | I_g(w)\right) \\ &= \prod_{s=1}^{S}\prod_{u=1}^{U_s} P_g(y^{(u)}(w)), \\ P_2 &= P(I_g(w)=1) = \pi_g, \end{aligned}$$

and

$$P_3 = \sum_{a=1}^{G} \pi_a \prod_{s=1}^{S}\prod_{u=1}^{U_s} P_a(y^{(u)}(w)).$$

As

$$P_a(y^{(u)}(w)) = \prod_{b=1}^{G}\{c_{ab}^{(s)}(\varphi)\}^{n_b^{(u)}(w)},$$

we have

$$\begin{aligned} \prod_{u=1}^{U_s} P_a(y^{(u)}(w)) &= \prod_{b=1}^{G}\{c_{ab}^{(s)}(\varphi)\}^{\sum_{u=1}^{U_s} n_b^{(u)}(w)} \\ &= \prod_{b=1}^{G}\{c_{ab}^{(s)}(\varphi)\}^{n_b^{(s)}(w)}. \end{aligned}$$

Therefore, we obtain the E-step in (31) by substituting the obtained expressions of P_1, P_2, and P_3.

The A-step (34) is the normalization of $h_g^{(\alpha)}(w)$ by $V^{(\alpha)}(w)$. We compute $h_g^{(\alpha)}(w)$ in (33) using (23), and we obtain

$$\boldsymbol{P}_g^{\psi} = \prod_{s=1}^{S}\prod_{b=1}^{G}\{c_{gb}^{(s)}(\psi)\}^{n_b^{(s)}(w)}.$$

$\boldsymbol{P}_g^{\varphi}$ has the same form as above except for ψ. As $V^{(\alpha)}(w)$ has the form in (32), we compute

$$V^{(\alpha)} = \sum_{g=1}^{G} h_g^{(\alpha)}(w).$$

Thus, we can compute the A-step in (34).

B. DERIVATION OF M-STEP FOR π_g^{ψ}

We start from the following differentiation for the maximization:

$$\frac{\partial}{\partial \pi_g^{\psi}}\left\{Q^{(\alpha)}_{\boldsymbol{z}|\boldsymbol{y},\varphi}(\psi|\varphi) + \lambda\left(\sum_{j=1}^{G}\pi_j^{\psi} - 1\right)\right\} = 0.$$

Then, we have

$$\frac{2}{1+\alpha}\frac{\partial S^{(\alpha)}_{z|y,\varphi}(\psi|\varphi)}{\partial \pi_g^{\psi}} + \lambda = 0 \tag{C.1}$$

because of (28). For the computation of $\frac{\partial S^{(\alpha)}_{z|y,\varphi}(\psi|\varphi)}{\partial \pi_g^{\psi}}$, we start from the logarithm of $S^{(\alpha)}_{\boldsymbol{z}|\boldsymbol{y},\varphi}(\psi|\varphi)$, which we denote by $S^{(\alpha)}$ here.

From (32), we have

$$\log S^{(\alpha)} = \sum_{w=1}^{W} \log V^{(\alpha)}(w).$$

Therefore, differentiating with respect to π_g^{ψ} and multiplying by $S^{(\alpha)}$ yield

$$\frac{\partial S^{(\alpha)}}{\partial \pi_g^{\psi}} = S^{(\alpha)} \sum_{w=1}^{W} \frac{1}{V^{(\alpha)}(w)} \frac{\partial V^{(\alpha)}(w)}{\partial \pi_g^{\psi}}.$$

One of the factors on the right-hand side is

$$\frac{\partial V^{(\alpha)}(w)}{\partial \pi_g^{\psi}} = \frac{1+\alpha}{2}\frac{h_g^{(\alpha)}(w)}{\pi_g^{\psi}}.$$

Therefore, by the definition of $\tilde{h}_g^{(\alpha)}(w)$ of (34), we have

$$\frac{\partial S^{(\alpha)}(w)}{\partial \pi_g^{\psi}} = \frac{1+\alpha}{2}\frac{S^{(\alpha)}}{\pi_g^{\psi}} \sum_{w=1}^{W} \tilde{h}_g^{(\alpha)}(w). \tag{C.2}$$

Then, by using (C.1) and (C.2), we have

$$S^{(\alpha)} \sum_{w=1}^{W} \tilde{h}_g^{(\alpha)}(w) + \lambda \pi_g^{\psi} = 0.$$

By summing over g and using (35), we have

$$\lambda = -S^{(\alpha)} W.$$

Therefore, we have the update equation (36).

C. DERIVATION OF M-STEP FOR $c_{gb}^{(s)}(\psi)$

Here, $c_{gb}^{(s)}(\psi)$ refers to the updated version of $c_{gb}^{(s)}(\varphi)$. We start from the following differentiation for maximization:

$$\frac{\partial}{\partial c_{gb}^{(s)}(\psi)}\left\{Q^{(\alpha)}_{\boldsymbol{z}|\boldsymbol{y},\varphi}(\psi|\varphi) + \lambda\left(\sum_{a=1}^{G} c_{ga}^{(s)}(\psi) - 1\right)\right\} = 0.$$

Then, we have

$$\frac{2}{1+\alpha}\frac{\partial S^{(\alpha)}(\psi|\varphi)}{\partial c_{gb}^{(s)}(\psi)} + \lambda = 0$$

from (28). By differentiating $\log S^{(\alpha)}$, we have

$$\frac{\partial S^{(\alpha)}}{\partial c_{gb}^{(s)}(\psi)} = S^{(\alpha)} \sum_{w=1}^{W} \frac{1}{V^{(\alpha)}(w)} \frac{\partial V^{(\alpha)}(w)}{\partial c_{gb}^{(s)}(\psi)}.$$

To compute the right-hand side of (37), we start from the differentiation of $\log V^{(\alpha)}(w)$. Then, we have

$$\begin{aligned} \frac{1}{V^{(\alpha)}(w)}\frac{\partial V^{(\alpha)}(w)}{\partial c_{gb}^{(s)}(\psi)} &= \frac{1+\alpha}{2}\tilde{h}_g^{(\alpha)}(w)\frac{\partial}{\partial c_{gb}^{(s)}(\psi)}\log \boldsymbol{P}_g^{\psi}(w) \\ &= \frac{1+\alpha}{2}\tilde{h}_g^{(\alpha)}(w)\frac{n_b^{(s)}(w)}{c_{gb}^{(s)}(\psi)}. \end{aligned}$$

Therefore, we have

$$S^{(\alpha)} \sum_{w=1}^{W} \tilde{h}_g^{(\alpha)}(w) n_b^{(s)} + \lambda c_{gb}^{(s)}(\psi) = 0. \tag{C.3}$$

By using $\sum_{b=1}^{G} c_{gb}^{(s)}(\psi) = 1$, we have

$$\begin{aligned} \lambda &= -S^{(\alpha)} \sum_{w=1}^{W}\sum_{b=1}^{G} \tilde{h}_g^{(\alpha)}(w) n_b^{(s)}(w) \\ &= -S^{(\alpha)} \sum_{w=1}^{W}\sum_{a=1}^{G} \tilde{h}_g^{(\alpha)}(w) n_a^{(s)}(w). \end{aligned} \tag{C.4}$$

By substituting (C.4) into (C.3) and by cancelling out $S^{(\alpha)}$, we have the update equation (37) in the main text.

APPENDIX D
PSEUDO CODE OF DYNAMIC-TUNING α-EM

We provide a simple code to understand the dynamic tuning α-EM quickly. Here, we suppressed various control keywords, such as while, continue, and break.

```
#define  W, S, G          // array sizes
#define  T, L             // constants
#define  EPSL, EPSD       // for convergence check
int      t_start, t_end;  // for clock()
short    table[W][S], nwsb[W][S][G], penalty[S];
int main() {
fopen input and output files;
prepare local variables in main();
```

```
/* data preparation */
read sommelier table as input data;
count nwsb[w][s][b];
multiple tasting penalty[s]++, if~exists;
input alpha_max, alpha_min;
input alpha_adj, start_alpha;
t_start=clock(); // clock start
intitial value assignment by (40);
/* main loop */
for(count=0; count<=T; count++) {
// initial cycle
if(count==0) use \text{log-EM} by (31), (36), (37);
/* \text{alpha-EM} starts from count==1 */
if(count>=1) {
compute \text{E-step} by (31);
compute lkhd[count] by (26);
/* convergence check if count>L */
if(count>L) {
compute lkhd_ratio by (41);
if(lkhd[count] decreased)
alpha=alpha_min;
compute max_pdist by (42);
convergence check by (41) and (42);
if( (convergence achieved) {
CONVCOUNT=count;
t_end=clock(); // clock end
compute sommelier evaluation;
penalize multiple tastings;
sorting by descending order;
print out results;
exit(0);
}
} // end of if(count>L)
compute \text{A-step} for count>=1 by (34);
compute \text{M-step} by (36) and (37);
increase alpha by (39);
} // end of each cycle of the alpha-EM
} // end of all iterations
// No convergence (this case did not occur)
printf("No convergence. T~is over!\n");
exit(0);
}
```

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.

[2] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[3] A. Rényi, "On measures of entropy and information," in *Proc. 4th Berkeley Symp. Math. Statist. Probab.*, vol. 1, 1961, pp. 547–561.

[4] S. M. Ali and S. D. Silvey, "A general class of coefficients of divergence of one distribution from another," *J. Roy. Stat. Soc., Ser. B Methodol.*, vol. 28, no. 1, pp. 131–142, Jan. 1966.

[5] I. Csiszár, "Information-type measures of difference of probability distributions and indirect observations," *Studia Sci. Math. Hungarica*, vol. 2, pp. 299–318, Jan. 1967.

[6] L. M. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Comput. Math. Math. Phys.*, vol. 7, no. 3, pp. 200–217, Jan. 1967.

[7] S. Amari and H. Nagaoka, *Methods Inf. Geometry*, Tokyo, Japan: Iwanami, 1993.

[8] Y. Matsuyama, "The α-EM algorithm: Surrogate likelihood maximization using α-logarithmic information measures," *IEEE Trans. Inf. Theory*, vol. 49, no. 3, pp. 692–706, Mar. 2003.

[9] Y. Matsuyama, "The alpha-HMM estimation algorithm: Prior cycle guides fast paths," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3446–3461, 2017.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm (with discussions)," *J. Roy. Statist. Soc. B Methodol.*, vol. 29, pp. 1–38, Sep. 1977.

[11] A. P. Dawid and A. M. Skene, "Maximum likelihood of observer error-rates using the EM algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 20–28, 1979.

[12] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, pp. 17545–17556, 2018.

[13] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, "Proof-of-Stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities," *IEEE Access*, vol. 7, pp. 85727–85745, 2019.

[14] Y. Matsuyama, "Divergence family attains blockchain applications via α-EM algorithm," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 727–731.

[15] R. Gray, A. Buzo, A. Gray, and Y. Matsuyama, "Distortion measures for speech processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 4, pp. 367–376, Aug. 1980.

[16] Y. Matsuyama and R. Gray, "Voice coding and tree encoding speech compression systems based upon inverse filter matching," *IEEE Trans. Commun.*, vol. 30, no. 4, pp. 711–720, Apr. 1982.

[17] B. D. Sharma and D. B. Mittal, "New non-additive measure of entropy for discrete probability distributions," *J. Math. Sci.*, vol. 10, pp. 28–40, 1975.

[18] S.-I. Amari, "α-divergence is unique, belonging to both F-divergence and bregman divergence classes," *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 4925–4931, Nov. 2009.

[19] F. Chollet, *Deep Learn. with Python*. Shelter Island, NY, USA: Manning Publications, 2018.

[20] Z. Thoutt. *Wine Reviews*. Accessed: Nov. 2018. [Online]. Available: https://www.kaggle.com/zynicide/wine-reviews/version/4

[21] H. Massias, S. Avila, and J.-J. Quisquater, "Timestamp: Main issues on their use and implementation," in *Proc. IEEE Int. Workshop Enabling Technol., Infrastruct. Collaborative Enterprises*, Stanford CA, USA, Jun. 1999, pp. 178–183.

[22] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[23] DeSieno, "Adding a conscience to competitive learning," in *Proc. IEEE Int. Conf. Neural Netw.*, San Diego CA, USA, 1988, pp. 117–124.

YASUO MATSUYAMA (Life Fellow, IEEE) received the B.Eng. and M.Eng. degrees and the Dr.Eng. degree in stochastic neural signal processing from Waseda University, Japan, in 1969, 1971, and 1974, respectively, and the Ph.D. degree in data compression theory and practice from Stanford University, Stanford, CA, USA, in 1978. He was granted a JSPS–Fulbright–IIE US–Japan Exchange Fellowship. Since 1996, he has been with Waseda University, where he is currently a Professor Emeritus and an Honorary Researcher. He was with the National Personnel Authority as a Co-Chairperson of the governmental personnel selection, in 1994. His research interests include statistical machine learning theory and its applications to the Internet of Collaborative Things. He is a Fellow of IEICE and IPSJ. He received the Outstanding Paper Award from the IEEE on the Transactions on Neural Networks, in 2001, and the Best Paper Award from the ACM-IEEE at the International Conference on Soft Computing as Transdisciplinary Science and Technology (CSTST), in 2008.