Modeling and forecasting overdispersed IoT data using an efficient and accurate computation of the Dirichlet Multinomial distribution

Sherenaz Al-Haj Baddar, Alessandro Languasco, Mauro Migliardi

Abstract— **The Internet of Things (IoT) has rapidly become a ubiquitous aspect of modern society, intertwining with our daily lives in many ways; while these advancements provide several beneficial effects, they also bring forth significant security concerns that must be addressed to safeguard sensitive data, ensure privacy, and prevent potential cyber threats. While security by design will undoubtedly help in hardening systems, security is and will remain an arm-race between attackers and defenders, hence, a most important feat in the field of security for networked systems is the capability to model traffic and provide predictions capable of identifying suspicious behavior. This task becomes more challenging when handling overdispersed data where the dataset variance is significantly greater than what is expected by the adopted model. In this paper we model the overdispersed traffic of data collected from a device belonging to an IoT system using the Dirichlet Multinomial distribution and use that model to forecast the device traffic for the next 24 hours. Also, we compare the efficiency and accuracy of some state of the art techniques in generating traffic predictions. In terms of building the forecast model, our experiments utilized the LM technique and showed its superiority against the widely-used YS technique, in terms of speed and model accuracy. Our experiments also showed that the forecast model we built was capable of accurately predicting the traffic forecast for the next three days.**

Index Terms— **Internet of Things, Forecast, Pattern analysis, overdispersed data, Dirichlet Multinomial distribution, log-likelihood function.**

I. Introduction

THE Internet of Things (IoT) has rapidly become a ubiquitous aspect of modern society, from smart home devices and wearable technology to industrial machinery and healthcare systems, IoT technologies offer immense convenience, efficiency, and connectivity. However, this widespread integration of IoT also brings forth significant security concerns that must be addressed to safeguard sensitive data, ensure privacy, and prevent potential cyber threats. As IoT devices continue to proliferate and collect vast amounts of personal and organizational data, the importance of security measures becomes increasingly paramount [13]. The interconnected nature of these devices creates an intricate web of vulnerabilities that malicious actors can exploit, leading to potential breaches, unauthorized access, and data manipulation. In fact, the implications of compromised IoT security extend beyond individual privacy concerns to encompass broader societal implications [12]. Vulnerabilities in critical infrastructure, such as energy grids, transportation systems, and healthcare facilities, could have far-reaching consequences, disrupting essential services and potentially endangering lives. Therefore, a robust approach to IoT security is imperative to mitigate risks and foster trust in the continued proliferation of IoT technologies in everyday life.

While ensuring robust security protocols is crucial to harden the system against possible attacks and safeguard the integrity of IoT ecosystems, it is not possible to achieve static security; security is and will remain an arm-race between attackers and defenders [9]. One technique that proved to be effective in this context is traffic forecast. A reliable model for predicting traffic shape and structure would trigger early-alarms when incoming traffic appears to deviate from the prediction. This would allow a sufficient time window for proactive countermeasures to be implemented. Besides, a robust framework for traffic predictions would also provide the basis for the optimizations of resources, something that is of paramount importance in a resource-constrained environment such as the IoT. However, when the traffic exhibits overdispersion characteristics, building a reliable forecast model that is efficiently computable becomes a daunting task. The aim of this study is to accurately and efficiently forecast overdispersed IoT traffic modeled using Dirichlet Multinomial (**DM**) distribution. To achieve this goal, historical IoT device traffic data was utilized to estimate the **DM** parameters and build the corresponding model that will be used to predict the traffic for the next 24-hours.

In our work, we adopted the dataset available at *https://data.mendeley.com/datasets/84cc8grtkt/1* that has the characteristic to be overdispersed. The Dirichlet Multinomial distribution plays a crucial role in modeling overdispersed count data by providing a flexible framework that captures the variability and dependencies among multiple counting variables. In situations where traditional models such as the Poisson or Multinomial distributions fail to account for the overdispersion observed in the data, the Dirichlet Multinomial distribution offers a more realistic and robust alternative. Overdispersion in count data arises when the variance of the observed counts exceeds the mean, indicating additional sources of variability or correlations that are not adequately captured by simpler models. To address this, we used the Dirichlet Multinomial distribution since it models the underlying distribution of probabilities of the counts. This allows for the incorporation of shared and individual parameters to account for both the variability within each count variable and the interdependence among them. By capturing the complex relationships and dependencies present in overdispersed count data, the Dirichlet Multinomial distribution enables more accurate inference, prediction, and modeling of such data. In this paper we adopt the fixed-point iteration

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

algorithm described in [8], and show how it is possible to generate accurate forecasts of the traffic parameters adopting the Dirichlet Multinomial distribution; furthermore, we evaluate different techniques performing such an estimation in terms both of time and accuracy. The paper is structured as follows, in Section II, we describe the estimation of the Dirichlet Multinomial distribution parameters. In Section III, we describe the experiments and their results, while some related work is summarized in Section IV. Finally, Section V provides some concluding remarks and highlights potential future work.

II. The Estimation of the Dirichlet Multinomial distribution parameters

The fixed-point iteration algorithm requires an initial guess for the he Dirichlet distribution parameter $\boldsymbol{\alpha}$. It also utilizes a maximum number of iterations and an upper limit (i.e., a threshold) on the acceptable difference between two consecutive estimates of $\boldsymbol{\alpha}$, denoted by Δ. In each iteration a new estimate for the parameter $\boldsymbol{\alpha}$, denoted by $\boldsymbol{\alpha}_1$, is generated. This is achieved by scanning the input data in order to calculate two paired differences of the digamma function as described in Eq. (55) in [8]. These two quantities are then used to estimate a new $\boldsymbol{\alpha}$ (i.e., $\boldsymbol{\alpha}_1$). Next, the absolute paired difference of the log-likelihood of the **DM** distribution at both $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ is calculated and compared to the threshold Δ; if that difference is less than Δ, then the algorithm is assumed to have converged. In this case, it returns $\boldsymbol{\alpha}_1$ and the accompanying parameter ψ_1. However, if no such convergence is reached after a certain number of iterations, the algorithm is assumed to have failed and the value $\boldsymbol{\alpha}_{\text{error}}$ is returned.

The soundness of the fixed-point iteration is jeopardized when the data is overdispersed since the paired difference comprises not one, but two calls to the **DM** $\log \mathscr{L}$ function. Each such call is prone to the lack-of-accuracy problem detailed in [5]. Two possible solutions are described in detail in [5] and [14].

III. Experimentation and Results

The experiments were carried out in two stages: the first aimed at building models that capture the nature of the device traffic using historical data, while the second aimed at evaluating the forecast accuracy of the models built in the first stage.

The fixed-point iteration technique described in [8] is employed in the first stage to estimate the parameters of the device traffic model. As stated in Section II, the difference between the log-likelihood of two consecutive estimates of the model parameter $\boldsymbol{\alpha}$ is used to decide if convergence is achieved.

In this context, two techniques to calculate this difference are contrasted in this stage, namely: **LM** [5], **YS** [14] with respect to speed and goodness of fit. To evaluate the speed of the modeling process, both the number of iterations and time to converge were evaluated. The goodness-of-fit, was assessed through several distance metrics that measure the deviation of the estimated model from the real one. These metrics include: Total Variation Distance (**tvd**), the Kullback–Leibler divergence Distance (**KL-D**), Euclidean Distance (**D**), and the Mean-Square-Error (**MSE**). Additionally, two statistical goodness-of-fit tests were applied to assess whether the estimated parameters produce a model close enough to the original. In the second stage, the forecast models generated in the first stage were compared to the actual models of each of the next 3 days. Here, we included Python's default implementation (**lgam-std**) of the log-likelihood function.

The experiments in this stage aimed at measuring forecasting speed, in terms of execution time, and accuracy in terms Brier Score (**BS**), and Jensen-Shannon Divergence (**JSD**). The results of the experiments of the first stage showed that **LM** outperformed **YS** in terms of all speed and goodness-of-fit metrics. The results of the second stage showed the superiority of **LM** in terms forecasting accuracy.

A. The Dataset

In the first stage of experiments, packet traces collected from an IoT device, namely Camera 5, of the D-Link IoT devices public dataset were aggregated into 10-minute count vectors, spanning seven workdays in the interval between Thursday, October 8$^{\text{th}}$, 2020, and Friday, October 16$^{\text{th}}$, 2020. Each vector featured the number of TCP (*T*ransmission *C*ontrol *P*rotocol), UDP (*U*ser *D*atagram *P*rotocol), SSDP (*S*imple *S*ervice *D*iscovery *P*rotocol), and ARP (*A*ddress *R*esolution *P*rotocol) packets observed during 10 minutes. In a similar fashion, packet traces from the days Monday, October 19$^{\text{th}}$, 2020, Wednesday, October 21$^{\text{st}}$, 2020, and Thursday, October 22$^{\text{nd}}$, 2020, were aggregated and used in the second stage of experiments. Table I summarizes the main aspects of the dataset used in this study which comprises 10 files (i.e., sub-datasets), each featuring the aggregates of a designated day. The sub-datasets corresponding to the seven days of traffic used in the first stage of experiments are referred to as "**training**" sub-datasets, whereas the ones used in the second stage are referred to as "**testing**" sub-datasets. In the context of this study, a dataset is overdispersed if $\psi < 0.05$. Thus, all sub-datasets employed in these experiments are overdispersed as illustrated in Table I.

Table I
The Sub-datasets

Sub-Dataset	Type	Instances	ψ
October 8$^{\text{th}}$	Training	196	0.004493
October 9$^{\text{th}}$	Training	143	0.004342
October 12$^{\text{th}}$	Training	143	0.003408
October 13$^{\text{th}}$	Training	78	0.003327
October 14$^{\text{th}}$	Training	143	0.008205
October 15$^{\text{th}}$	Training	143	0.004914
October 16$^{\text{th}}$	Training	143	0.006185
October 19$^{\text{th}}$	Testing	124	0.008430
October 21$^{\text{st}}$	Testing	143	0.004099
October 22$^{\text{nd}}$	Testing	143	0.007148

As described in Algorithm 1 in [8], the fixed-point iteration continued until either the maximum number of iterations was reached, or until the difference between two consecutive estimates for $\boldsymbol{\alpha}$ was less than the tolerance value Δ. Equation (1) illustrates the formula used for calculating Δ:

$$\Delta := \frac{10^{-\texttt{prec}-2}}{K+1}, \tag{1}$$

where `prec` denotes the level of accuracy of the log-likelihood value using the **LM** procedure, in terms of the decimal digits of the mantissa, and K designates the number of categories. Accordingly, the value of Δ in this study is equal to $2.00 \cdot 10^{-9}$. The number of categories, K, is equal to 4 for all "training" and "testing" sub-datasets.

B. Experimentation Setup

A script that implements the fixed-point iteration algorithm was developed using both **LM** and **YS** in Python (3.10.2) with numpy (1.22.2), mpmath (1.2.1), and scipy (1.8.0) packages. The script also utilized Python's math `math.lgamma` function in order to calculate the **lgam-std** log-likelihood values. The paired differences of the F function in the original Algorithm in [8] were computed using the C-compiled `digamma(x)` function implementation provided in the scipy package. The experiments were executed on an Intel Core i5 computer with 8 GB of RAM. The Euclidean distance metric, denoted by **D**, was calculated using the `LA.norm` function from the numpy package, whereas the χ^2-test was calculated using the scipy.stats `chisquare` function. Each test was repeated three times and its results were averaged.

C. The First Stage of Experimentation: Modeling

In the first stage of experiments, a **DM** model for each "training" sub-dataset was built using random samples of each of the designated "training" sub-datasets with varying sizes: 10%, 15%, 20%, and 25%. The number of iterations and time to converge corresponding to the **LM** and **YS** techniques[1], are depicted in Figures 1 and 2, respectively. Next, the goodness-of-fit metrics, illustrated in Table II, and the χ^2-test and Z-test were applied using only the **LM** technique, since the **YS** technique failed the δ-barrier test several times, which hindered the accuracy of its estimates. The results are depicted in Tables III and IV, respectively.

1) Iterations to converge: Figure 1 compares **LM** and **YS** in terms of the number of iterations to converge (denoted by IC), and shows that the numbers of iterations consumed by both techniques are rather close. For example, the traces collected on October 9th and 13th, took the most IC, in the order of 20k, to converge for both techniques when 10% of the sub-dataset was used. On the other hand, these two sub-datasets required almost 7K and 13K iteration to converge for 25% of the input, respectively. It is worth noticing that almost 65% of the **YS** tests depicted in Fig.1-3, failed due to the δ-barrier problem described in [1, §II-A]. This resulted in by-passing this issue to get the code to execute and return an estimate of the corresponding model parameters. As Figure 1 shows, the number of iterations per se is not enough to differentiate the performance, and a further metric is required. Next, we contrast the performance of these techniques using the time to converge, i.e., execution speed.

2) Time to Converge: Figure 2 shows that the **LM** technique significantly outperforms **YS** in terms of the time to converge, denoted by TC. For each sub-dataset and across all sample sizes, it is obvious that the **LM** converged much faster than **YS**. It is worth noticing that **YS** took extremely longer to converge. For example, **YS** needed almost 8 minutes to converge for the October 13th sub-dataset with a 20% sample size, while it took **LM** only 6 seconds using the exact input. To further illustrate the superiority of **LM** over **YS** in terms of time, Figure 3 depicts the speedup **LM** achieved over **YS**, for each of the "training" sub-datasets. The highest speed up was achieved for the October 13th subset at 20% of the input size, which was about 80. If this extreme speedup is excluded, the average speedup in Fig. 3 is 14. The lowest speedup, which is equal to 9.11, was reported for the October 8th subset at 20% of the input.

3) Goodness of Fit: A set of distance metrics was employed to evaluate the goodness of the estimated $\boldsymbol{\alpha}$ and ψ **DM** parameters, that includes: Total Variation Distance (**tvd**), the Kullback–Leibler divergence Distance (**KL-D**), Euclidean Distance (**D**), and the Mean-Square-Error (**MSE**). Table II illustrates the distance metrics evaluated in this study.

Where:

- P and Q denote the real and expected probability vectors, respectively;

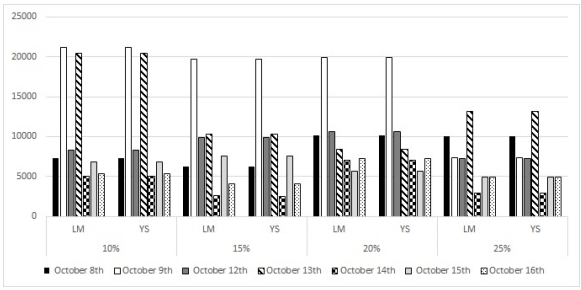

Figure 1. Percentage of Dataset Utilized Vs. The Number of Iterations to Converge (IC) for **LM** and **YS** (seconds).

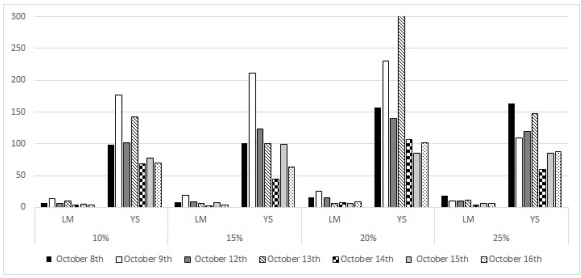

Figure 2. Percentage of Dataset Utilized Vs. The Time to Converge (TC) for **LM** and **YS** (seconds).

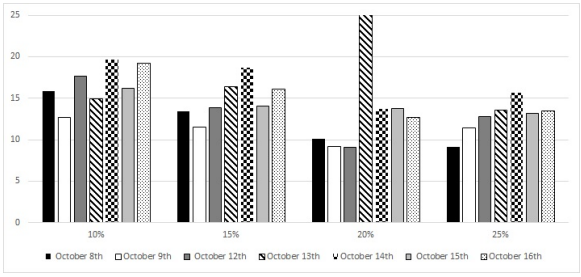

Figure 3. Percentage of Dataset Utilized Vs.The Speedup of **LM** over **YS**.

[1]In this stage, **lgam-std** was not included as its speed and model goodness-of-fit are comparable to those of **LM**.

Table II
DISTANCE AND GOODNESS-OF-FIT FUNCTIONS

Function	Description
Total Variation Distance (**tvd**)	$\textbf{tvd}(P,Q) := \frac{1}{2}\sum_{k=1}^{K} \lvert P[k]-Q[k]\rvert$
KL-Distance (**KL-D**)	$\textbf{KL-D}(P,Q) := \sum_{k=1}^{K}(P[k]\cdot\log(\frac{P[k]}{Q[k]})$
Euclidean Distance (**D**)	$\textbf{D}(P,Q) := (\sum_{k=1}^{K}(P[k]-Q[k])^2)^{1/2}$
Mean-Square-Error (**MSE**)	$\textbf{MSE} := \frac{1}{K}\sum_{k=1}^{K}(P[k]-Q[k])^2$
Chi-squared Test (χ^2-test)	$\chi^2\text{-test}\ (O,E) := \sum_{k=1}^{K}\frac{(O_k-E_k)^2}{E_k}$
The Z-test (Z-test)	$Z\text{-test}\ (O,E) := \frac{\overline{x}-\mu}{\sigma}$

- O and E denote the real (i.e., observed) and expected counts vectors, respectively;
- $\overline{x}$ denotes the mean of the expected counts vector, while μ denotes the mean of the real counts vector, and σ denotes its standard deviation.

The distance functions in Table II were computed for each of the experiments of the first stage resulting in the estimates illustrated in Table III. The table shows that the **tvd** values ranged between 0.004393 and 0.053707, while the **KL-D** values ranged between 0.000073 and 0.012963. Table III also shows that the values of **D** ranged between 0.005111 and 0.062229, while the values of **MSE** ranged between 0.000007 and 0.000984. The results of applying the χ^2-test and Z-test are presented in Table IV, and they confirm that the estimated models managed to capture the real ones, as both tests were passed in all 14 experiments, except for one.

Table III
ERROR ESTIMATES IN ψ

Dataset	**tvd**	**KL-D**	**D**	**MSE**
October 8th	0.053707	0.012963	0.062229	0.000984
October 9th	0.009424	0.000551	0.011645	0.000039
October 12th	0.008008	0.000309	0.009234	0.000023
October 13th	0.004393	0.000073	0.005111	0.000007
October 14th	0.011128	0.000776	0.013238	0.000052
October 15th	0.009806	0.000533	0.011855	0.000037
October 16th	0.013229	0.001066	0.016108	0.000072

Table IV
THE STATISTICAL TESTS χ^2-TEST AND Z-TEST RESULTS

Dataset	χ^2-test	Z-test
October 8th	fail	pass
October 9th	pass	pass
October 12th	pass	pass
October 13th	pass	pass
October 14th	pass	pass
October 16th	pass	pass
October 16th	pass	pass

D. The Second Stage of Experimentation: Forecasting

In the second stage of experimentation, the models developed in the first stage for each of the "training" sub-datasets were sampled 1000 times, and these samples were averaged resulting in a prediction (i.e., forecast) model that was used to predict the traffic for the next 24 hours. In this stage, the three techniques: **LM**, **YS**, and **lgam-std** were employed to build the prediction models. To evaluate the forecast accuracy, Brier Score (**BS**) and Jensen-Shannon Divergence (**JSD**) were employed. More precisely, these two metrics were used to measure the distance between the model predicted by **LM**, **YS**, and **lgam-std**, and the real model for each of the three "testing" sub-datasets. The

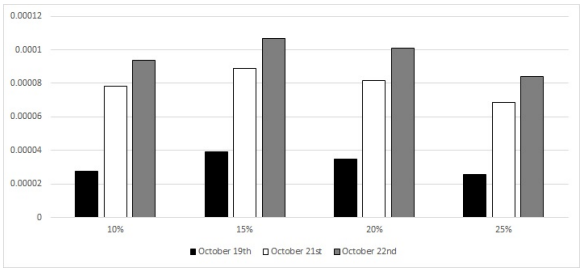

Figure 4. Percentage of Dataset Utilized Vs.The Brier Scores using the **LM** prediction model for the testing subsets.

results of initial experiments, as illustrated in Figure 4, show that the prediction accuracy increases as the size of the input used in building the prediction model increases. Thus, both **BS** and **JSD** were measured using models built with 25% of the input. Table V shows the time it took the designated techniques to produce their models for the seven training sub-datasets in the experiments carried out with 25% of the input sizes to measure the **BS** and **JSD** values. It shows the superiority of **lgam-std** and **LM** compared to **YS** in terms of speed as its TC was more than 20-fold that of the other two techniques. Table V also shows that the TC of **LM** is within 10% of **lgam-std**, thus, there is no significant difference in speed between both techniques. It is worth noticing that unlike **LM** which is developed purely in Python, the **lgam-std** loggamma function is C-optimized. Therefore, **LM** has the potential to execute significantly faster than **lgam-std** in case its Python code was C-optimized, too. As Table VI shows, the best **BS** for almost all the "testing" sub-dataset was achieved by the **LM** technique. Moreover, the smallest **BS** value is $1.11 \cdot 10^{-5}$ achieved for the October 19th sub-dataset. Similarly, **LM** achieved the best **JSD** values for all three "testing" sub-datasets; the smallest **JSD** value is $1.65 \cdot 10^{-5}$ achieved also for the October 19th sub-dataset. Table VI shows in a bold-face the best **BS** and **JSD** for each "testing" sub-dataset.

Table V
TC VALUES FOR **LGAM-STD**, **LM**, AND **YS**

Technique	$TC(s)$
lgam-std	10.923
LM	11.083
YS	264.075

Table VI
SCORES

	LM		**lgam-std**		**YS**	
	BS	**JSD**	**BS**	**JSD**	**BS**	**JSD**
October 19th	$\mathbf{1.11 \cdot 10^{-5}}$	$\mathbf{1.65 \cdot 10^{-5}}$	$1.36 \cdot 10^{-5}$	$1.99 \cdot 10^{-5}$	$1.24 \cdot 10^{-5}$	$1.78 \cdot 10^{-5}$
October 21st	$5.09 \cdot 10^{-5}$	$\mathbf{6.70 \cdot 10^{-5}}$	$5.31 \cdot 10^{-5}$	$7.13 \cdot 10^{-5}$	$\mathbf{4.98 \cdot 10^{-5}}$	0.000197
October 22nd	$\mathbf{6.30 \cdot 10^{-5}}$	$\mathbf{9.09 \cdot 10^{-5}}$	$6.72 \cdot 10^{-5}$	$9.74 \cdot 10^{-5}$	$6.34 \cdot 10^{-5}$	0.000148

IV. RELATED WORK

Forecasting count data by building next-day prediction models has been explored in recent studies. For example, the study in [2] introduced a novel class of dynamic state-space models for univariate count time series data. Experiments, in the context of consumer sales forecasting, showed the superiority

of the proposed models when compared to baseline models, in terms of scaled mean squared error (sMSE), mean absolute deviation (MAD), and mean ranked probability score (MRPS). The work in [4] proposed a Bayesian optimization-based dynamic ensemble, that combined forecasts from ten different models (statistical, machine learning, and deep learning). combination for better adaptability and generalization. The model was compared to state-of-art models using datasets with seasonal patterns that illustrated its superiority. A similar study is depicted in [7], where a probabilistic forecasting method for hourly-load time series based on an improved temporal fusion transformer (ITFT) model was introduced. The model utilized both statistical and machine learning techniques in order to build compact prediction models and learn long-term dependence efficiently. The work in [11], introduced an autoregressive recurrent neural network model trained on several time series. Experiments showed the superiority of this approach compared to traditional methods, and with less manual setup. A similar approach was employed in [10] which proposed an autoregressive model for multivariate probabilistic time series forecasting. This model sampled from the data distribution by estimating its gradient which the model learned by optimizing a variational bound on the data likelihood. The study compared the performance of the proposed model to other state-of-the-art models, and showed its superiority using several real-world datasets. The reader is referred to [6], [15], and [3] for other recent studies.

V. Conclusions and Future work

In this study, we have compared the computational efficiency of different methodologies to compute the Dirichlet Multinomial distribution and we have tested its efficacy in modeling and forecasting overdispersed IoT network traffic data. The technique is based on the maximization of the **DM** log-likelihood function using the fixed-point iteration algorithm presented in [8]. More in details, we have evaluated three different methodologies to compute the paired differences of the loggamma function values that are used in the cited technique. The three methodologies evaluated are the ones described in [5] (**LM**), [14] (**YS**) and the default approach (**lgam-std**) that uses the functions of the `math` package released with the Python programming language. The modeling experiments were used to compare the performances of the **LM** and **YS** techniques in terms of time to converge, number of iterations, and goodness of fit, and established the superiority of **LM**. Another set of experiments compared the three techniques in terms of forecasting efficiency and accuracy, and showed the superiority of **LM** with respect to **YS** for both characteristics and, in terms of accuracy, with respect to the **lgam-std** technique. Regarding efficiency, we have to remark that the difference between the execution time observed for **LM** and the one for **lgam-std** was always less than a 10% factor. However, the **LM** technique is implemented purely in Python while the core of the `math` package used in **lgam-std** is a Python wrapping of a highly optimized C-code. Thus, there is room for improvement by porting **LM** to the C language and wrapping it in Python. Another aspect to be explored is the impact of employing approaches other than the fixed-point iteration to help estimate the **DM** model parameters and compare their performances in terms of efficiency and accuracy.

References

[1] S. Al-Haj Baddar, A. Languasco, M. Migliardi, "Efficient analysis of overdispersed data using an accurate computation of the Dirichlet Multinomial distribution", in IEEE Transactions on Pattern Analysis and Machine Intelligence, DOI:*https://doi.org/10.1109/TPAMI.2024.3489645*.

[2] L.R. Berry, M. West, "Bayesian Forecasting of Many Count-Valued Time Series", *J. Bus. Econ. Stat.* **38** (4), 872–887 (2019). DOI:*https://doi.org/10.1080/07350015.2019.1604372*.

[3] Y. Chen, Y. Kang, Y. Chen, Z. Wang, "Probabilistic forecasting with temporal convolutional neural network", *Neurocomputing* **399**, 491–501 (2020). DOI:*https://doi.org/10.1016/j.neucom.2020.03.011*.

[4] L. Du, R. Gao, P.N. Suganthan, D.Z.W. Wang, "Bayesian optimization based dynamic ensemble for time series forecasting", *Inf. Sci.* **591**, 155–175 (2022). DOI:*https://doi.org/10.1016/j.ins.2022.01.010*.

[5] A. Languasco, M. Migliardi, "On the fast computation of the Dirichlet-multinomial log-likelihood function", *Comput. Stat.* **38**, 1995–2013 (2023). DOI:*https://doi.org/10.1007/s00180-022-01311-7*.

[6] V. Le Guen, N. Thome, "Probabilistic time series forecasting with structured shape and temporal diversity", in *Proc. 34th International Conference on Neural Information Processing Systems (NIPS '20)*, paper no. 372, 4427–4440 (2020). *https://cedric.cnam.fr/ thomen/papers/STRIPE-NeurIPS'20.pdf*.

[7] D. Li, Y. Tan, Y. Zhang, S. Miao, S. He, "Probabilistic forecasting method for mid-term hourly load time series based on an improved temporal fusion transformer model", *Int. J. Elec. Power* **146**, paper no. 108743, 2023. DOI:*https://doi.org/10.1016/j.ijepes.2022.108743*.

[8] T.P. Minka, "Estimating a Dirichlet distribution", 2000 (revised 2003, 2009, 2012), *https://tminka.github.io/papers/dirichlet/minka-dirichlet.pdf*.

[9] P. Nespoli, D. Papamartzivanos, F. Gómez Màrmol and G. Kambourakis, "Optimal Countermeasures Selection Against Cyber Attacks: A Comprehensive Survey on Reaction Frameworks", *IEEE Commun. Surv. Tut.* **20** (2), 1361–1396 (2018). DOI:*https://doi.org/10.1109/COMST.2017.2781126*.

[10] K. Rasul, C. Seward, I. Schuster, R. Vollgraf, "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting", *International Conference on Machine Learning*, PLMR 139, 8857–8868 (2021). *https://proceedings.mlr.press/v139/rasul21a/rasul21a.pdf*.

[11] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks", *Int. J. Forecast.* **36** (3), 1181–1191, (2020). DOI:*https://doi.org/10.1016/j.ijforecast.2019.07.001*.

[12] S.W. Smith and E.H. Spafford, "Grand challenges in information security: process and output", *IEEE Security & Privacy* **2** (1), 69–71, (2004). DOI:*https://doi.org/10.1109/MSECP.2004.1264859*.

[13] S. Vashi, J. Ram, J. Modi, S. Verma and C. Prakash, "Internet of Things (IoT): A vision, architectural elements, and security issues", *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, 2017, pp. 492–496. DOI:*https://doi.org/10.1109/I-SMAC.2017.8058399*.

[14] P. Yu, C.A. Shaw, "An efficient algorithm for accurate computation of the Dirichlet Multinomial log-likelihood function", *Bioinformatics* **30**, 1547–1554, (2014). DOI:*https://doi.org/10.1093/bioinformatics/btu079*.

[15] Z. Zhang, C. Wang, X. Peng, H. Qin, Hui, H. Lv, J. Fu, H. Wang, "Solar Radiation Intensity Probabilistic Forecasting Based on K-Means Time Series Clustering and Gaussian Process Regression", *IEEE Access* **9**, 89079–89092, (2021). DOI:*https://doi.org/10.1109/ACCESS.2021.3077475*.

[16] B. Zhao, Y. Zhong, GS. Xia, L. Zhang, "Dirichlet-derived multiple topic scene classification model for high spatial resolution remote sensing imagery", *IEEE Trans. Geosci. Remote Sens.* **54** (4), 2108–2023, (2015). DOI:*https://doi.org/10.1109/TGRS.2015.2496185*.

Sherenaz Al-Haj Baddar, The University of Jordan, Department of Computer Science, Amman 11942, Jordan. e-mail: `s.baddar@ju.edu.jo`

Alessandro Languasco, Università di Padova, Dipartimento di Ingegneria dell'Informazione, Via Gradenigo 6/b, 35131 Padova, Italy. e-mail: `alessandro.languasco@unipd.it`

Mauro Migliardi, Università di Padova, Dipartimento di Ingegneria dell'Informazione, Via Gradenigo 6/b, 35131 Padova, Italy. e-mail: `mauro.migliardi@unipd.it`