Incremental Parameter Estimation of Stochastic State-Based Models

Robert Lipp*, Guido Dartmann†, Lejla Fazlic†, Thomas Vollmer‡, Stefan Winter‡,
Arne Peine§, Lukas Martin§ and Anke Schmeink*
*Research Area ISEK, RWTH Aachen University, Germany
Email: {lipp, schmeink}@isek.rwth-aachen.de
†Environmental Campus, Trier University of Applied Sciences, Germany
Email: {g.dartmann, l.begic}@umwelt-campus.de
‡Philips Research, Aachen, Germany
Email: {thomas.vollmer, stefan.winter}@philips.com
§Department of Intensive Care Medicine and Intermediate Care, University Hospital RWTH Aachen, Germany
Email: {apeine, lmartin}@ukaachen.de

***Abstract*—This paper presents an incremental learning approach for estimating the structural parameters in stochastic state-based models (SSMs). SSMs have proven to be useful for modelling biological and medical processes, as they can represent both time dependency and stochastic processes. A major challenge in modelling in bioinformatics is that learning processes usually rely on large publicly accessible databases. In this work, a new approach is presented, where models are trained incrementally locally at different data sources, e.g., hospitals, without having to pass on sensitive data. After learning, only the parameters of the model are passed on, in this case the arc weights of stochastic Petri nets. As a result, data protection and privacy of patients in hospitals are respected and it is no longer necessary to rely on the existence of a suitable accessible database. Simulations are used to evaluate the performance of the algorithm for a gene regulatory network.**

***Index Terms*—Bioinformatics, incremental learning, federated learning, stochastic state-based models**

I. Introduction

Sepsis is the leading cause of death in intensive care worldwide [1]. The diagnosis and treatment of sepsis is a challenging task for physicians.

For this reason, a lot of research has been done recently on automatic recognition and therapy management. Due to many vital signs and laboratory values available from patients in intensive care units, this can be considered a typical task for machine learning. A prominent example is the Artificial Intelligence Clinician presented by Komorowski et al. [1].

The usual procedure for obtaining data is that individual projects collect data from selected hospitals, process it, make it anonymous and then publish it. The probably best known such published database is MIMIC-III [2]. The publication of such databases makes it possible to find machine learning solutions for medical applications, which require a large amount of training data. However, being dependent on such databases also has disadvantages. On the one hand, they do not necessarily contain enough different patients, so that cohorts with special required characteristics are simply too small. On the other hand, there may be shortcomings in the quality and quantity of the entries. Furthermore, the data is not always sorted and standardised to the desired extent. This can result in the data required for a particular task being entered into the database at insufficient frequency or in an inappropriate manner. Another major problem is that the EU's General Data Protection Regulation [3] makes it difficult to publish such databases containing sensitive data.

In this paper, we therefore use the approach of federated learning, which has been explored intensively in recent times and is not based on publicly accessible databases. A detailed overview of federated learning was presented by Yang et al. [4]. The authors state that they expect that the smart healthcare sector will benefit greatly from federated learning. Google proposed federated learning as an approach to build models using machine-learning on data that is distributed among different devices [5]–[7], which demonstrates the relevance of this approach also to the Internet of Things. In [8], a system is presented which allows several parties to learn a neural network without sharing the input data. In a series of publications, Ohno-Machado and her team have shown that federated learning is also ideally suited for medical applications taking into account privacy and data security aspects [9]–[13].

In this work, we have adapted the concept of federated learning to our needs and present a new algorithm for learning the structural parameters of a stochastic state-based model in the biomedical domain using distributed data sets. The general procedure is as follows. The first step is to determine what data is required for a particular task and in what form it should be available. This data is then collected directly in the hospitals in the required manner and used to train a model. Unfortunately, as mentioned before, high data protection and privacy regulations make this practically impossible, as patient data cannot simply be shared. For this reason, we present a procedure that makes the sharing of patient data no longer necessary by only using measured data locally in one hospital

This project was funded by the Federal Ministry of Education and Research (BMBF) grant 13GW0280A, 13GW0280C, 13GW0280D, and 13GW0280E as part of the IMEDALytics project.

and not transferring it. To achieve this, a mathematical model is trained in a hospital. Then, only the parameters of the model are passed on, so that no conclusions can be drawn about the sensitive patient data.

Of course, this procedure is highly dependent on the mathematical model that is used. It has been shown that stochastic state-based models are very well suited for biological and medical processes, since they can represent time-dependent processes on the one hand and take into account the stochastic nature of the processes under consideration on the other hand [14].

The remainder of this paper is organized as follows. In Section II, the stochastic state-based model used in this paper is mathematically defined. In Section III, the iterative learning algorithm is introduced in three different variants. In Section IV, the third and most general variant of the iterative algorithm is evaluated using simulations. In Section V, the simulation results are briefly discussed. Section VI concludes this work and gives an outlook on future research.

II. Stochastic State-Based Model

The stochastic state-based model (SSM) used in this work is similar to a classical stochastic Petri net (SPN) [15], which means that it is a bipartite graph with two types of nodes: Places and transitions. Each place contains a non-negative number of tokens while transitions move tokens from their input places to their output places. The delay between enabling and firing of a transition is exponentially distributed.

In medical applications, Petri nets can be used in two main ways. The first application is the modelling of states, e.g. to represent the condition of a patient. The Places each represent a state, while existing tokens indicate in which state the patient is currently. The changes of states are then modeled using transitions. In the second possible application, each place stands for a substance, while the number of tokens in a particular place indicates the concentration of that substance. The transitions of the Petri net then represent chemical reactions in which educts are converted into products.

In the following, we will formally define SSMs. The places are denoted by p_i, $1 \leq i \leq P$, and the transitions are denoted by τ_j, $1 \leq j \leq T$. The weight of the arc connecting p_i with τ_j is denoted as $[\mathbf{Pre}]_{i,j}$ while the arc from τ_j to p_i has weight $[\mathbf{Post}]_{i,j}$. The incidence matrix is defined as $A = \mathbf{Post} - \mathbf{Pre} \in \mathbb{N}_0^{P \times T}$. It is possible that in a SPN a place is both input and output place of a transition, in this case the SPN is called impure and the matrix A does not describe the SPN uniquely. For this reason, in this work we assume that the SPN does not have this property and is therefore pure. The number of tokens contained in a place p_i is denoted as $m(p_i)$ and the total marking is described by the vector $[m(p_1), \ldots, m(p_P)]^{\mathrm{T}}$.

In the following, we address the following challenge. The goal is to estimate the entries of A, i.e., the arc weights of the SPN. The given data is a set of L measurements where for each $1 \leq \ell \leq L$ a series of noisy markings $(\tilde{m})_k^{(\ell)} = \tilde{m}_1^{(\ell)}, \ldots, \tilde{m}_{K_\ell}^{(\ell)}$ is given. To consider data protection and privacy, we aim at the learning process to be iterative, which means it consists of L independent learning steps where in step ℓ only the data $(\tilde{m})_k^{(\ell)}$ is available.

An important preliminary work is an algorithm called likelihood-based decision-aided adaptive gradient descent (LB-DAAGD) proposed in [16] and [17] that solves the problem for $L = 1$. For $L > 1$ an iterative learning approach must be used. For this purpose the algorithm mentioned above will be adapted.

III. Iterative Algorithm

The algorithm taken from [17] is based on a state space formulation. Given the matrix A that describes the topology of the SPN, we consider the state equation

$$m_{k+1} = m_k + A \cdot u_k \tag{1}$$

where m_k is the state at step k, $u_k \in \{0,1\}^T$ is the firing vector that states which transition fires at step k and m_{k+1} is the new state of the system. The goal of the algorithm is to find an estimation of matrix A. Starting from an initial matrix $\hat{A}_0$, this gradient descent algorithm computes a sequence of approximations $\hat{A}_k$ of the matrix A on which the system is based. The mean square error of the predicted status at the current time serves as the objective function. The update equation then looks like this:

$$\hat{A}_{k+1} = \hat{A}_k - \mu \cdot (\hat{m}_{k+1} - \tilde{m}_{k+1}) \cdot u_k^T. \tag{2}$$

In each update step, only the current and subsequent state of the system are required. We want to exploit this iterative nature of the algorithm in the following.

For this purpose we consider several hospitals that want to jointly train a model of a biological system. However, due to data protection regulations and the need to protect the privacy of patients, no patient data may be exchanged between hospitals. Classical machine learning approaches that require all training data at once during the training phase would not succeed in this task. In the following, we present three concepts how the mentioned incremental algorithm can be used for our application.

A. Sequential Approach

In the use case of training models in hospitals, the iterative approach is as follows: A hospital collects data from patients and locally trains a mathematical model. The final model and the errors measured during the training phase are passed on to the next hospital. It is then no longer possible to draw conclusions about the data originally collected in the first hospital, which means that the data protection and privacy of the patients are fulfilled. Now, the second hospital can start training the model further. The reason that this is possible is because the algorithm in [17] works incrementally by nature and does not, like other approaches, need all training data at once. The general procedure is shown in Figure 1.

Formally, initially for $\ell = 1$ the LB-DAAGD algorithm is executed using the data $(\tilde{m})_k^{(1)}$. The result is an estimation $\hat{A}^{(1)}$ of the incidence matrix A of the SSM. Note that the dimension of $\hat{A}^{(1)}$ is not necessarily $(P \times T)$, as the algorithm

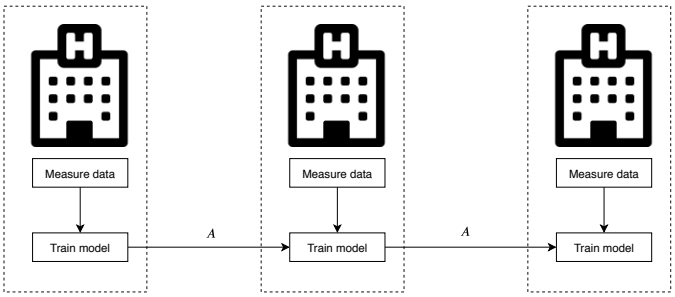

Fig. 1. Incremental learning of a model A in different hospitals

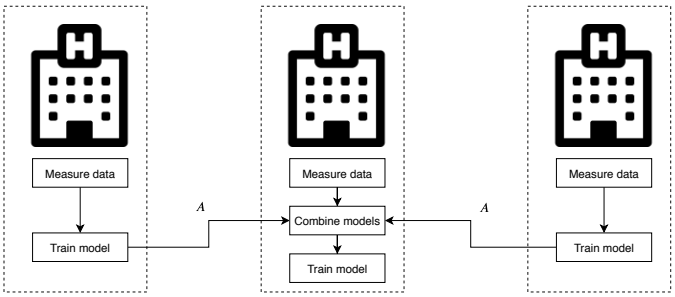

Fig. 2. Incremental learning in different hospitals with parallel training

may not detect some transitions or may detect some transitions more than once. Next, the algorithm is executed again with data $(\tilde{m})_k^{(2)}$, where $\hat{A}^{(1)}$ is used as the start matrix and so on. Finally, a matrix $\hat{A}^{(L)}$ is computed, which has been trained with all data without needing all data at once. A pseudo code is shown in Algorithm 1.

Algorithm 1: Incremental Parameter Estmation: Sequential

Result: Matrix $\hat{A}^{(L)}$
$\hat{A}^{(0)}$ = empty matrix;
for $\ell \in \{1, \dots, L\}$ **do**
  $\hat{A}^{(\ell)}$ = Result of LB-DAAGD starting with $\hat{A}^{(\ell-1)}$ and data $(\tilde{m})_k^{(\ell)}$
end

B. Sequential Approach with Parallel Learning

An advantage of the sequential approach is that it is very simple and already implicitly contained in the LB-DAAGD. One drawback is that parallel learning is not possible. If two hospitals train a model at the same time, from which hospital would a third hospital request the model to continue training it? To solve this problem, we present a method to combine several models into one. The general procedure is shown in Figure 2.

First, we show how to find similar transitions in a model and merge them into a single one.

1) Merging similar transitions: In the learning process it can happen that a transition is recognized more than once due to inaccurate data or poorly selected learning parameters. When joining several models, it can also be useful to first copy the learned transitions from all models. As a result, matrix $\hat{A}$ may contain several columns describing the same transition. Afterwards similar columns can be merged into a single column. We consider two columns $v_1, v_2 \in \mathbb{R}^P$ to be similar if $||v_1 - v_2||_\infty < d$ for some parameter $d > 0$. The parameter d depends on the underlying system. For example, if the entries of A are known to be integers, $d = 1 - \varepsilon$ for a small ε is a reasonable choice, because two different transitions v_1, v_2 should differ in at least one component, which implies $||v_1 - v_2||_\infty \geq 1$.

As soon as a model now contains similar transitions, i.e., the corresponding matrix A contains a set S of similar columns, these can be removed and replaced their average $\frac{1}{|S|}\sum_{v' \in S} v'$.

2) Merging different models: If several already trained models are available, it makes sense to first combine them into a single model and then to continue training this model. A way to do this is to arrange all the columns of the corresponding matrices into a single large matrix. If the different models were all similarly well trained, then each transition should be represented by multiple columns. These columns should be similar in the sense of the similarity defined above and can therefore be merged. In this way, a model is created in which each transition occurs only once and can now be trained with the new data. A pseudo code for merging several models is shown in Algorithm 2.

Algorithm 2: Merge different models

Result: Matrix $\hat{A}$
Input: Different models $\hat{A}_1, \dots, \hat{A}_k$;
$\hat{A}$ = concatenate($\hat{A}_1, \dots, \hat{A}_k$);
while $\hat{A}$ *contains similar columns* **do**
  S = set of similar columns of $\hat{A}$;
  remove all columns in S from $\hat{A}$;
  add $\frac{1}{|S|}\sum_{v' \in S} v'$ to $\hat{A}$;
end

C. Central Storage Approach

The approaches presented in III-A and III-B provide the ability to train a model without a large public training data set. To this end, the model was transferred to hospitals and trained locally, taking data protection and privacy into account. However, there are also aspects that have not been considered so far. How can it be ensured that the model does not deteriorate during training? How can it be guaranteed that a model is not trained several times with the same data? These problems can be solved through supervised incremental learning. For this purpose, a central storage unit manages the model. When a hospital has collected new data, it requests the current model, trains it, and returns the modified model and information about the measured error values during the training phase. An overview of the procedure is shown in Figure 3. This approach offers many advantages. On the one hand, the central unit can carry out version control. This

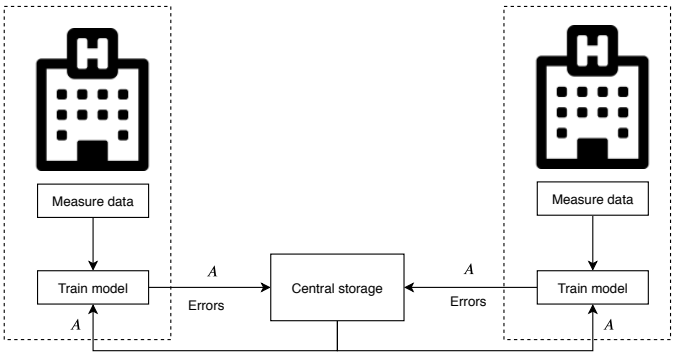

Fig. 3. Incremental learning in different hospitals with central storage

ensures that it is always clear which model was trained with which data. Another great advantage is that it is possible to check how the models develop over time by monitoring the error values of the single training phases. In a further step, the current model could be sent to all participating hospitals to check how well it matches all previously collected data. Another approach to ensure the quality of the models could be to carry out anomaly detection to check, for example, whether individual hospitals provide bad data.

IV. Simulation Results

In this section, we validate the central storage approach using simulations. As described in the previous section, this approach is the most promising and also the most general of the presented approaches, since the central memory can manage the trained models in any desired way. In the following simulations, the central memory unit will store all trained models as a list and always output the most recent model for training. In fact, this procedure effectively is the same as the first approach plus version control.

We use a predefined stochastic Petri net, which was already used in [17] for algorithm evaluation. The stochastic Petri net originally comes from [14] and models a circadian clock model. Although the actual use case for our algorithm is supposed to be sepsis, in this early phase of algorithm development it makes sense to investigate a well-known Petri net instead of the not yet fully understood process of sepsis. We carry out a total of three different simulation and training settings. Each setting is run 50 times to hide the effects of the stochastic nature of the model. Setting A simulates the case that the algorithm is executed in a single hospital with sufficient data volume. In setting B we simulate that in a single hospital not enough data is available. Setting C deals with the case that the individual hospitals have measured exactly the same amount of data as in setting B, but this time the model is trained incrementally in 20 hospitals one after the other.

A. Enough training data

In this subsection, we consider the case where there is enough training data to train the model sufficiently well for $L = 1$, so there is just one training step. A run of the algorithm consists of the following steps:

1) Simulate the Petri net until 5000 firings of tranisitions occured.
2) Train the model with these data.
3) Calculate coefficient Mean Square Error (MSE) and the number of detected transitions of each learning step.

Fig. 4 shows the results for this case.

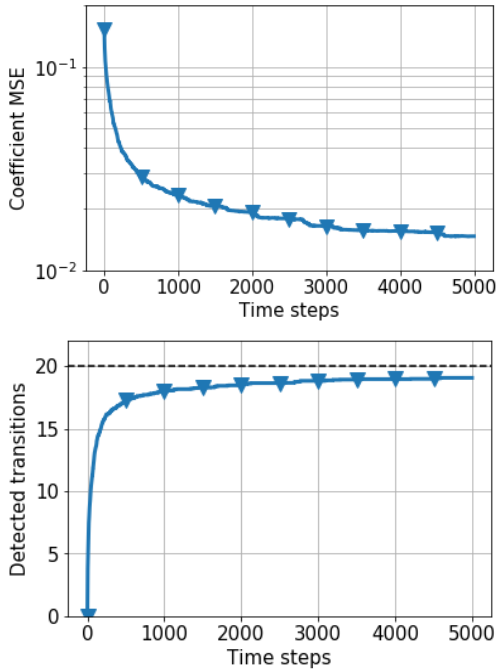

Fig. 4. Training with a sufficient amount of 5000 data points

B. Insufficient data

In this subsection, we consider the case where there is insufficient training data to train the model sufficiently well for $L = 1$. This setting corresponds to the case that too little suitable data is available in a publicly accessible database. Again we simulate the given Petri net, but this time only until 250 data points are simulated.

Simulations have shown that due to the fact that the simulation always starts with the same initial marking m_0, the markings $m_0, \ldots, m_{249}$ are quite similar for the different runs. This is because with this small number of firings the stochastic nature of the model does not come into play enough. For this reason, we ran two different simulations: One was using the markings $m_0, \ldots, m_{249}$ and the other was using the markings $m_{n-250}, \ldots, m_{n-1}$ for a randomly chosen $n > 250$.

A run of the algorithm basically consists of the same steps as in setting 1. The results are shown in Fig. 5.

C. Incremental training with small batches

In this subsection, we use the new approach of incremental learning. For this purpose, 20 batches are generated in independent simulations from 250 data points each. The model is then trained with the individual batches one after the other, whereby it only has access to one batch at a time. This

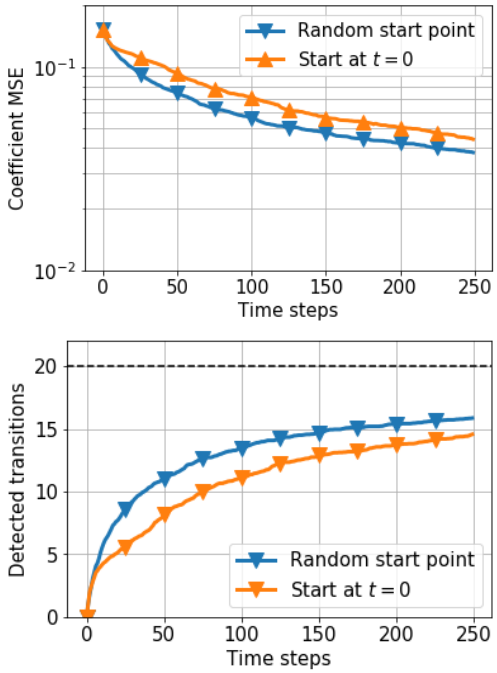

Fig. 5. Training with an insufficient amount of 250 data points

corresponds to local learning in a hospital without passing on the data. The results are shown in Fig. 6. The dashed orange lines show the steps where a new batch is used.

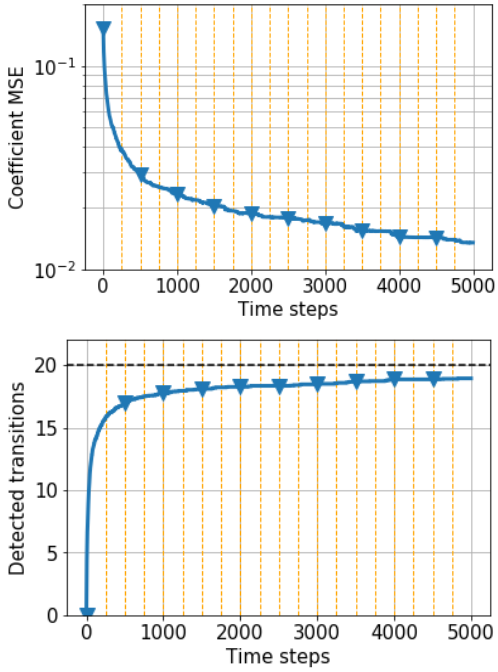

Fig. 6. Incremental training with 20 batches of 250 data points each

V. Discussion

In this section, we briefly discuss the simulation results presented in Section IV. Fig. 4 shows the results of the algorithm being executed with a sufficient amount of 5000 data points. The mean squared error of the entries of the reconstructed incidence matrix converges to a low level and the algorithm detects around 19 of the 20 transitions of the underlying Petri net in average. The fact that not all transitions are recognized in all runs can be explained by the fact that there are single transitions in the Petri net that almost never fire and therefore cannot be detected reliably.

However, the plots presented in Fig. 5 show that the algorithm cannot reliably reconstruct the model if there are too few data points, which was to be expected. Although the MSE is falling, it is not converging noticeably. In addition, several transitions are not detected because they fire barely or not at all in the too short observed time period. This behavior is typical for machine learning algorithms that receive too little training data.

The results presented in Fig. 6 were obtained through the incremental approach. Compared to Fig. 4 there is barely any difference in the plots. This clearly shows that incremental learning in our case with several small data sets works just as well as model learning with one large data set.

The fact that Fig. 4 and 6 do not show exactly the same plots is due to the fact that the data were simulated stochastically for each training. Simulating the Petri nets 50 times eliminates the significant outliers, but does not produce exactly the same result.

VI. Conclusion

In this paper, we have presented a method to recover the parameters of stochastic Petri nets in an incremental way from measured data. Our main contributions were: (1) We have extended the algorithm from [17] and adapted it for incremental applications, (2) we have developed several designs of the algorithm, and (3) we have shown through simulations that the most general version of our incremental algorithm performs as well as the algorithm that has a large database available. More specifically, on the one hand, the simulations have shown that too small data sets alone are not sufficient to train the model sufficiently well for the case that only one data source is available. On the other hand, it is sufficient to use several small data sets incrementally one after the other for training. In this way, the learning algorithm is no longer dependent on large publicly accessible data sets. Instead, the required data can be collected and used locally, for example in hospitals, without violating privacy and data protection. Future research includes further testing of the algorithm with patient data measured in hospitals. Furthermore, it is an open problem how to estimate the kinetic parameters of SPN incrementally based on measured data.

References

[1] M. Komorowski, L. Celi, O. Badawi, A. Gordon, and A. Faisal, "The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care," *Nature Medicine*, vol. 24, 11 2018.

[2] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, p. 160035, 2016.

[3] EU, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance)," 2016. [Online]. Available: https://op.europa.eu/s/n7mV

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, Jan. 2019. [Online]. Available: https://doi.org/10.1145/3298981

[5] J. Konecný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *CoRR*, vol. abs/1610.02527, 2016. [Online]. Available: http://arxiv.org/abs/1610.02527

[6] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016.

[7] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016.

[8] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1310–1321. [Online]. Available: https://doi.org/10.1145/2810103.2813687

[9] S. Wang, X. Jiang, Y. Wu, L. Cui, S. Cheng, and L. Ohno-Machado, "Expectation propagation logistic regression (explorer): Distributed privacy-preserving online model learning," *Journal of Biomedical Informatics*, vol. 46, no. 3, pp. 480 – 496, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1532046413000427

[10] Z. Ji, X. Jiang, S. Wang, L. Xiong, and L. Ohno-Machado, "Differentially private distributed logistic regression using private and public data," *BMC Med Genomics*, vol. 7, 2014. [Online]. Available: https://doi.org/10.1186/1755-8794-7-S1-S14

[11] C.-L. Lu, S. Wang, Z. Ji, Y. Wu, L. Xiong, X. Jiang, and L. Ohno-Machado, "WebDISCO: a web service for distributed cox model learning without patient-level data sharing," *Journal of the American Medical Informatics Association*, vol. 22, no. 6, pp. 1212–1219, 07 2015. [Online]. Available: https://doi.org/10.1093/jamia/ocv083

[12] K. K. Kim, D. K. Browe, H. C. Logan, R. Holm, L. Hack, and L. Ohno-Machado, "Data governance requirements for distributed clinical research networks: triangulating perspectives of diverse stakeholders," *Journal of the American Medical Informatics Association*, vol. 21, no. 4, pp. 714–719, 12 2013. [Online]. Available: https://doi.org/10.1136/amiajnl-2013-002308

[13] J. Que, X. Jiang, and L. Ohno-Machado, "A Collaborative Framework for Distributed Privacy-Preserving Support Vector Machine Learning," *AMIA Annu Symp Proc. 2012*, p. 1350–1359, 2012. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3540462/

[14] M. A. Blätke, M. Heiner, and W. Marwan, *BioModel Engineering with Petri Nets*, 05 2015, pp. 141–192.

[15] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, 2nd ed. Springer Publishing Company, Incorporated, 2010.

[16] P. Vieting, R. C. de Lamare, G. Dartmann, and A. Schmeink, "An adaptive learning approach to parameter estimation for hybrid petri nets in systems biology," in *IEEE Statistical Signal Processing Workshop (SSP)*, Freiburg, Germany, Jun. 2018, pp. 1–6. [Online]. Available: http://www.ti.rwth-aachen.de/publications/output.php?id=1131&table=proceeding&type=pdf

[17] P. Vieting, R. C. de Lamare, L. Martin, G. Dartmann, and A. Schmeink, "Likelihood-based adaptive learning in stochastic state-based models," *IEEE Signal Processing Letters*, pp. 1–5, May 2019. [Online]. Available: http://www.ti.rwth-aachen.de/publications/output.php?id=252&table=article&type=pdf