



For simplicity, $m(\mathbf{x})$ is usually assumed to be zero. Then, we choose the squared exponential function, defined by

$$c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \theta_0^2 \exp \left\{ -\frac{1}{2} \sum_{d=1}^D \frac{(x_d - x'_d)^2}{\theta_d^2} \right\}, \quad (2)$$

as the kernel function, where θ_d , $d = 0, 1, \dots, D$ are positive hyperparameters that are optimized in the training process. More details about covariance functions can be found in Rasmussen and Williams [1].

Suppose that we have a training dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where y_n is obtained by adding i.i.d. Gaussian noise, subject to $\mathcal{N}(0, \sigma^2)$, to $f_n = f(\mathbf{x}_n)$. Let \mathbf{X} , \mathbf{f} , and \mathbf{y} denote all training inputs, all corresponding latent function values, and all training outputs, respectively. Then, the training process is performed by maximizing the log-likelihood function, given by

$$L(\mathbf{y}; \boldsymbol{\theta}, \sigma) = \frac{1}{N} \log p(\mathbf{y}) = \frac{1}{N} \log \mathcal{N}(\mathbf{y} | 0, \mathbf{C}_{NN} + \sigma^2 \mathbf{I}_N), \quad (3)$$

w.r.t. $\boldsymbol{\theta}$ and σ , where $\mathbf{C}_{NN} = c(\mathbf{X}, \mathbf{X}; \boldsymbol{\theta})$ and \mathbf{I}_N is the identity matrix.

After the training process, given a test point (\mathbf{x}^*, y^*) , the aim of the prediction process is to calculate the conditional distribution $p(y^* | \mathbf{y})$. We have

$$\begin{pmatrix} \mathbf{y} \\ y^* \end{pmatrix} \sim \mathcal{N} \left(0, \begin{pmatrix} \mathbf{C}_{NN} + \sigma^2 \mathbf{I}_N & \mathbf{c}_{*N}^T \\ \mathbf{c}_{*N} & c_{**} + \sigma^2 \end{pmatrix} \right), \quad (4)$$

where $\mathbf{c}_{*N} = c(\mathbf{x}^*, \mathbf{X}; \boldsymbol{\theta})$ and $c_{**} = c(\mathbf{x}^*, \mathbf{x}^*; \boldsymbol{\theta})$. It follows that

$$y^* | \mathbf{y} \sim \mathcal{N} \left(\mathbf{c}_{*N} (\mathbf{C}_{NN} + \sigma^2 \mathbf{I}_N)^{-1} \mathbf{y}, c_{**} + \sigma^2 - \mathbf{c}_{*N} (\mathbf{C}_{NN} + \sigma^2 \mathbf{I}_N)^{-1} \mathbf{c}_{*N}^T \right) \quad (5)$$

From Eq. (3–5), we see that the time complexity of training GPs scales as $\mathcal{O}(N^3)$ and the space complexity as $\mathcal{O}(N^2)$, since we need to store $\mathbf{C}_{NN} + \sigma^2 \mathbf{I}_N$ and calculate its inverse and determinant. That makes GPs intractable for large datasets.

Next, we shortly introduce the SGP model that can overcome the above limitation. M inducing points $\{(\mathbf{z}_m, u_m)\}_{m=1}^M$ are introduced to construct an SGP, where \mathbf{z}_m , $m = 1, \dots, M$ are pseudo-inputs independent of \mathbf{X} , and $u_m = f(\mathbf{z}_m)$. Let \mathbf{Z} and \mathbf{u} be all the pseudo-inputs and all inducing variables, respectively. Then, it is obtained that

$$\begin{aligned} L(\mathbf{y}; \boldsymbol{\theta}, \sigma) &= \frac{1}{N} \log p(\mathbf{y}) \\ &= \frac{1}{N} \log \int p(\mathbf{u}, \mathbf{f}, \mathbf{y}) d\mathbf{f} d\mathbf{u} \\ &= \frac{1}{N} \log \int q(\mathbf{u}, \mathbf{f}) \frac{p(\mathbf{u}) p(\mathbf{f} | \mathbf{u}) p(\mathbf{y} | \mathbf{f})}{q(\mathbf{u}, \mathbf{f})} d\mathbf{f} d\mathbf{u} \\ &\geq \frac{1}{N} \int q(\mathbf{u}, \mathbf{f}) \log \frac{p(\mathbf{u}) p(\mathbf{f} | \mathbf{u}) p(\mathbf{y} | \mathbf{f})}{q(\mathbf{u}, \mathbf{f})} d\mathbf{f} d\mathbf{u} \end{aligned} \quad (6)$$

in which $q(\mathbf{u}, \mathbf{f})$ is any probability distribution over $(\mathbf{u}; \mathbf{f})$, and the inequality is obtained through Jensen's inequality. The coefficient $1/N$ is used to eliminate the impact of the scale of the gradients. Assume that $q(\mathbf{u}, \mathbf{f}) = q(\mathbf{u})p(\mathbf{f}|\mathbf{u})$, where $q(\mathbf{u})$ is an unconstrained Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. It follows that

$$\begin{aligned} L(\mathbf{y}; \boldsymbol{\theta}, \sigma) &\geq F(\boldsymbol{\theta}, \sigma, \mathbf{Z}, q(\mathbf{u})) \\ &= \frac{1}{N} \int q(\mathbf{u})p(\mathbf{f}|\mathbf{u}) \log \frac{p(\mathbf{u})p(\mathbf{y}|\mathbf{f})}{q(\mathbf{u})} d\mathbf{f}d\mathbf{u}. \end{aligned} \quad (7)$$

Fixing $\boldsymbol{\theta}, \sigma$ and \mathbf{Z} , $q^*(\mathbf{u})$ that maximizes $F(\boldsymbol{\theta}, \sigma, \mathbf{Z}, q(\mathbf{u}))$ can be found analytically. The mean vector and covariance matrix of $q^*(\mathbf{u})$ are

$$\boldsymbol{\mu}^* = \frac{1}{\sigma^2} \mathbf{C}_{MM} \mathbf{A}^{-1} \mathbf{C}_{MN} \mathbf{y} \text{ and } \boldsymbol{\Sigma}^* = \mathbf{C}_{MM} \mathbf{A}^{-1} \mathbf{C}_{MM}, \quad (8)$$

respectively, where $\mathbf{C}_{MM} = c(\mathbf{Z}, \mathbf{Z}; \boldsymbol{\theta})$, $\mathbf{C}_{MN} = c(\mathbf{Z}, \mathbf{X}; \boldsymbol{\theta})$, and $\mathbf{A} = \mathbf{C}_{MM} + \sigma^{-2} \mathbf{C}_{MN} \mathbf{C}_{MN}^T$. Then, we have

$$\begin{aligned} L(\mathbf{y}; \boldsymbol{\theta}, \sigma) &\geq F(\boldsymbol{\theta}, \sigma, \mathbf{Z}) = F(\boldsymbol{\theta}, \sigma, \mathbf{Z}, q^*(\mathbf{u})) \\ &= \frac{1}{N} \log \mathcal{N}(\mathbf{y} | 0, \mathbf{Q}_{NN} + \sigma^2 \mathbf{I}_N) - \frac{1}{2N\sigma^2} \text{tr}(\mathbf{C}), \end{aligned} \quad (9)$$

where $\mathbf{Q}_{NN} = \mathbf{C}_{MN}^T \mathbf{C}_{MM}^{-1} \mathbf{C}_{MN}$ and $\mathbf{C} = \mathbf{C}_{NN} - \mathbf{C}_{MN}^T \mathbf{C}_{MM}^{-1} \mathbf{C}_{MN}$. Next, the estimation of $\boldsymbol{\theta}$ and σ by maximizing $L(\mathbf{y}; \boldsymbol{\theta}, \sigma)$ is replaced with the joint estimation of $\boldsymbol{\theta}, \sigma$, and \mathbf{Z} by maximizing $F(\boldsymbol{\theta}, \sigma, \mathbf{Z})$. This replacement enables the reduction in the time and space complexity.

After the above maximization, we can calculate an approximation of the true conditional distribution $p(\mathbf{y}^*|\mathbf{y})$. We have

$$p(\mathbf{y}^*|\mathbf{y}) = \int p(\mathbf{u}|\mathbf{y})p(\mathbf{f}|\mathbf{u}, \mathbf{y})p(\mathbf{y}^*|\mathbf{u}, \mathbf{f})d\mathbf{f}d\mathbf{u}. \quad (10)$$

By substituting $p(\mathbf{u}|\mathbf{y})$ with $q^*(\mathbf{u})$ and $p(\mathbf{y}^*|\mathbf{u}, \mathbf{f})$ with $p(\mathbf{y}^*|\mathbf{u})$, we obtain an approximate distribution

$$q(\mathbf{y}^*) = \int q^*(\mathbf{u})p(\mathbf{y}^*|\mathbf{u})d\mathbf{u}. \quad (11)$$

$q(\mathbf{y}^*)$ is a Gaussian distribution, whose mean and variance are

$$m_{\mathbf{y}}(\mathbf{x}^*) = \frac{1}{\sigma^2} \mathbf{c}_{*M} \mathbf{A}^{-1} \mathbf{C}_{MN} \mathbf{y} \quad (12)$$

and

$$c_{\mathbf{y}}(\mathbf{x}^*) = c_{**} + \sigma^2 - \mathbf{c}_{*M} \left(\mathbf{C}_{MM}^{-1} - \mathbf{A}^{-1} \right) \mathbf{c}_{*M}^T, \quad (13)$$

respectively, where $\mathbf{c}_{*M} = c(\mathbf{x}^*, \mathbf{Z}; \boldsymbol{\theta})$.

3 Federated Sparse Gaussian Processes

3.1 FederatedAveraging Algorithm

Suppose that there are K clients and the k th one possesses a local training dataset $\mathcal{D}_k = \{(\mathbf{x}_n^k, y_n^k)\}_{n=1}^{N_k}$, $k = 1, \dots, K$. Furthermore, assume that $\mathcal{D} = \cup_{k=1}^K \mathcal{D}_k$ and $N = \sum_{k=1}^K N_k$. To conduct federated learning, we use a factorized target function w.r.t. clients to approximate the true likelihood, i.e. $p(\mathbf{y}) \approx \prod_{k=1}^K p(\mathbf{y}_k)$, which leads to $\log p(\mathbf{y}) = \sum_{k=1}^K \log p(\mathbf{y}_k)$. This approximation has been applied to the training of distributed GPs [21, 22].

As shown in Sect. 2, $1/N_k \log p(\mathbf{y}_k)$ has a lower bound $F_k(\boldsymbol{\theta}, \sigma, \mathbf{Z})$, which is defined on \mathcal{D}_k in the way shown in Eq. (9). $F_k(\boldsymbol{\theta}, \sigma, \mathbf{Z})$, $k = 1, \dots, K$ have common parameters. Therefore, $\sum_{k=1}^K N_k/NF_k(\boldsymbol{\theta}, \sigma, \mathbf{Z})$ can be viewed as an approximate lower bound of $1/NF(\boldsymbol{\theta}, \sigma, \mathbf{Z})$. The form of this lower bound is similar to that of the objective function of the federated optimization problem in McMahan et al. [16]. Thus, we use the FederatedAveraging algorithm proposed by McMahan et al. [16] to train an FSGP. Algorithm 1 gives the local update processes on clients.

A

In the two algorithms, \mathbf{w} and \mathbf{w} represent $\{\boldsymbol{\theta}, \sigma, \mathbf{Z}\}$ for simplicity. Since $F_k(\boldsymbol{\theta}, \sigma, \mathbf{Z})$ has the coefficient $1/N_k$, it is rational to consider the scales of the gradients of $F_k(\boldsymbol{\theta}, \sigma, \mathbf{Z})$, $k = 1, \dots, K$ to be same. Thus, we use the same learning rate sequence for different clients. To improve the training efficiency, only $\max\{K\rho, 1\}$ clients are selected to update model parameters locally in one round, where $\rho \in (0, 1)$. In addition, we can employ privacy-preserving techniques, such as fully homomorphic encryption [23, 24], to ensure data security when transmitting gradients [18].

3.2 Prediction

After an FSGP is trained through the above FederatedAveraging algorithm, we can use Eq. (12) and Eq. (13) to calculate the approximate predictive distribution $q(\mathbf{y}^*)$. To show that the calculation can preserve privacy, we rewrite Eq. (12) and Eq. (13) as

$$m_{\mathbf{y}}(\mathbf{x}^*) = \frac{1}{\sigma^2} \mathbf{c}_{*M} \left(\mathbf{C}_{MM} + \frac{1}{\sigma^2} \sum_{k=1}^K \mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T \right)^{-1} \left(\sum_{k=1}^K \mathbf{C}_{MN_k} \mathbf{y}_k \right) \quad (14)$$

and

$$c_{\mathbf{y}}(\mathbf{x}^*) = c_{**} + \sigma^2 - \mathbf{c}_{*M} \left(\mathbf{C}_{MM}^{-1} - \left(\mathbf{C}_{MM} + \frac{1}{\sigma^2} \sum_{k=1}^K \mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T \right)^{-1} \right) \mathbf{c}_{*M}^T, \quad (15)$$

respectively, where $\mathbf{C}_{MN_k} = c(\mathbf{Z}, \mathbf{X}_k; \boldsymbol{\theta})$. From Eq. (14) and Eq. (15), we see that if a client wants to calculate $q(\mathbf{y}^*)$, it solely needs the values of $\mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T$ and $\mathbf{C}_{MN_k} \mathbf{y}_k$ from the other clients. Since \mathcal{D}_k cannot be recovered from the values of $\mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T$ and $\mathbf{C}_{MN_k} \mathbf{y}_k$ (see Theorem 1), the prediction is privacy-preserving.

Theorem 1. \mathcal{D}_k cannot be recovered from the values of $\mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T$ and $\mathbf{C}_{MN_k} \mathbf{y}_k$.

Proof. Since an input \mathbf{x} and a pseudo-input \mathbf{z}_m are both real vectors, it is rational to consider that \mathbf{x} is impossible to be equal to \mathbf{z}_m . Thus, any entry of \mathbf{C}_{MN_k} belongs to the open interval $(0, \theta_0^2)$. View each row of \mathbf{C}_{MN_k} as a point in $(0, \theta_0^2)^{N_k}$. $(0, \theta_0^2)^{N_k}$ is an open set and the convex hull \mathcal{C} of the M points is a subset of it. It follows that there exist infinitely many rotation transformations around the origin, denoted as φ , so that $\varphi(\mathcal{C})$ is still a subset of $(0, \theta_0^2)^{N_k}$. Each φ can be regarded as an $N_k \times N_k$ orthogonal matrix \mathbf{Q}_φ . Then, we have

$$\mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T = (\mathbf{C}_{MN_k} \mathbf{Q}_\varphi) (\mathbf{C}_{MN_k} \mathbf{Q}_\varphi)^T \quad (16)$$

and

$$\mathbf{C}_{MN_k} \mathbf{y}_k = (\mathbf{C}_{MN_k} \mathbf{Q}_\varphi) (\mathbf{Q}_\varphi^T \mathbf{y}_k) \quad (17)$$

Therefore, we cannot infer \mathbf{C}_{MN_k} and \mathbf{y}_k from the values of $\mathbf{C}_{MN_k} \mathbf{C}_{MN_k}^T$ and $\mathbf{C}_{MN_k} \mathbf{y}_k$. Then, that \mathbf{C}_{MN_k} cannot be recovered leads to that \mathbf{X}_k cannot be recovered. We can easily generalize this result to other covariance functions.

4 Experiments

In this section, we present the experimental results on two synthetic datasets and one real-world dataset. The first dataset is drawn from the following function of one variable

$$f(x) = 3\sin(2\pi x/20), x \in [-10, 10]. \quad (18)$$

The 500 training inputs are evenly distributed in the above interval and corresponding outputs are obtained by adding i.i.d. Gaussian noises, subject to $\mathcal{N}(0, 0.5^2)$, to latent function values. The 300 test samples are generated in the same way. The second synthetic dataset is generated similarly. The latent function is

$$f(\mathbf{x}) = 2.5\sin(2\pi(x_1 + x_2)/90), \mathbf{x} \in [-25, 25]^2 \quad (19)$$

This dataset consists of 4900(70×70) training samples and 900(30×30) test samples. The Gaussian noises follow $\mathcal{N}(0, 0.4^2)$. The third dataset is KIN40K dataset, which contains 10000 training samples and 30000 test samples from $\mathbb{R}^8 \times \mathbb{R}$.

We use the root mean squared error (RMSE) to measure the performance of SGPs, FGPs and FSGPs, which is defined as

$$\text{RMSE} = \sqrt{\frac{1}{L} \sum_{l=1}^L (t_l - y_l)^2} \quad (20)$$

where $\{y_l\}_{l=1}^L$ and $\{t_l\}_{l=1}^L$ are test outputs and corresponding predictions, respectively. It is clear that smaller RMSE imply better performance.

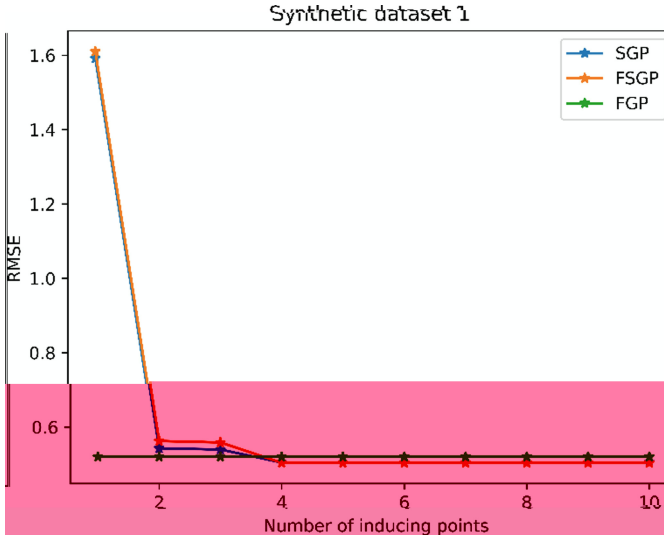


Fig. 1. Synthetic dataset 1

In all the three experiments, T , P and λ are set to be 5000, 3 and 0.1, respectively. Then, we sequentially set $K = 5, 10, 10$ and $K\rho = 2, 5, 5$, respectively. θ , σ , and \mathbf{Z} are

initialized as $(1, \dots, 1)^T$, 0.1, and a random subset of \mathbf{X} , respectively. When training SGPs and FSGPs, θ , σ , and \mathbf{Z} have the same initial values. Furthermore, the imbalance problem is considered in the experiments by randomly determining the sizes of training subsets. In the first experiment, the difference between the maximum number and the minimum one is 59. In the other two experiments, the differences are 634 and 1299, respectively.

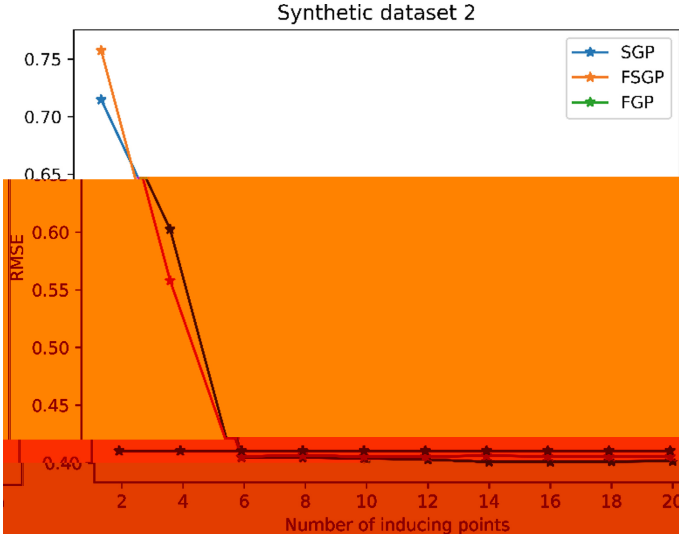


Fig. 2. Synthetic dataset 2

The results on three datasets are presented in Fig. 1, Fig. 2, and Fig. 3, respectively. In all the three experiments, FSGPs perform better than FGPs. On the two synthetic datasets, FSGPs outperform FGPs slightly when the number of inducing variables is large enough. However, on the KIN40K dataset, FSGPs obviously outperform FGPs when the number of inducing variables is large enough, since the unknown latent function in KIN40K is more complex than the two synthetic latent functions. In addition, we see that FSGPs and SGPs have a similar ability, that is to say, FSGPs

are comparable with SGPs. The three results show that although the whole training datasets are divided into small subsets in training an FSGP, we can obtain comparable performance through the federated aggregation algorithm.

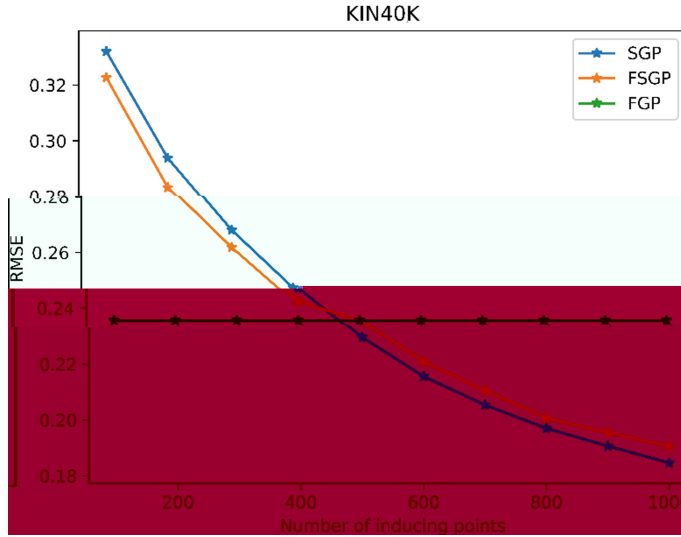


Fig. 3. KIN40K dataset

5 Conclusion

We have proposed an FSGP model that not only remains the scalability of SGPs, but also can learn a shared model using isolated datasets stored on more than one client. The FSGP model can preserve privacy since, in the training process, we need not transport the data stored on one client to the other clients, and in the test process, the data cannot be recovered. The experimental results on two synthetic datasets and one real-world dataset show that the performance of our proposed FSGP model is comparable with that of SGPs and better than that of FGPs in terms of the criterion we adopt. Two interesting topics for the future is to develop a more effective algorithm to accelerate the training processes and to combine vertical federated learning with GPs.

Acknowledgement. This work is supported by the National Key R & D Program of China (2018AAA0100205).

References

1. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press (2006)
2. Bishop, C.M. (ed.): Pattern Recognition and Machine Learning. ISS, Springer, New York (2006). <https://doi.org/10.1007/978-0-387-45528-0>
3. Guo, X., Li, X., Ma, J.: Variational EM algorithm for Student- mixtures of Gaussian processes. In: Huang, D.S., Jo, K.H., Li, J., Gribova, V., Hussain, A.: (eds) Intelligent Computing Theories and Application. ICIC 2021. Lecture Notes in Computer Science, vol. 12837, pp. 552–563. Springer, Cham (2021) https://doi.org/10.1007/978-3-030-84529-2_47

4. Gao, X.B., Wang, X.M., Tao, D.C.: Supervised Gaussian process latent variable model for dimensionality reduction. In: IEEE transactions on systems, man, and cybernetics, Part B (Cybernetics), vol. 41, no. 2, pp. 425–434 (2011)
5. Smola, A.J., Bartlett, P.: Sparse greedy Gaussian process regression. In: Advances in Neural Information Processing System, vol. 13, pp. 619–625 (2001)
6. Csato, L., Opper, M.: Sparse online Gaussian processes. In: Neural Computation, vol. 14, pp. 641–648 (2002)
7. Lawrence, N.D., Seeger, M., Herbrich, R.: Fast sparse Gaussian process methods: the informative vector machine. In: Advances in Neural Information Processing Systems, vol. 15 (2003)
8. Schwaighofer, A., Tresp, V.: Transductive and inductive methods for approximate Gaussian process regression. In: Advances in Neural Information Processing Systems, vol. 15 (2003)
9. Seeger, M., Williams, C.K.I., Lawrence, N.D.: Fast forward selection to speed up sparse Gaussian process regression. In: Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics, PRML R4, pp. 254–261 (2003)
10. Williams, C.K.I., Seeger, M.: Using the Nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems, vol. 13, MIT Press (2001)
11. Quinonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate Gaussian process regression. In: Journal of Machine Learning Research, vol. 6, pp. 1939–1959 (2005)
12. Snelson, E., Ghahramani, Z.: Sparse Gaussian process using pseudo-inputs. In: Advances in Neural Information Processing Systems, vol. 18 (2006)
13. Titsias, M.K.: Variational model selection for sparse Gaussian process regression. Technical report, School of Computer Science, University of Manchester (2009)
14. Titsias, M.K.: Variational learning of inducing variables in sparse Gaussian processes. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, PMLR 5, pp. 567–574 (2009)
15. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. In: ACM Transactions on Intelligent Systems and Technology, vol. 10 (2019)
16. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR 54, pp. 1273–1282 (2017)
17. Yue X., Kontar R.A.: Federated Gaussian Process: Convergence, Automatic Personalization and Multi-Fidelity Modeling. <https://doi.org/10.48550/arXiv.2111.14008>
18. Yang, Q., Liu, Y., Cheng, Y., Kang, Y., Chen, T., Yu, H.: Federated Learning (2019)
19. Cheng, K., et al.: Secureboost: a lossless federated learning framework. In: IEEE Intelligent Systems (2021)
20. Zhuo, H.H., Feng, W., Lin, Y., Xu, Q., Yang, Q.: Federated Deep Reinforcement Learning. arXiv.org (2020–02–09) <https://arxiv.org/abs/1901.08277>
21. Deisenroth, M.P., Ng, J.W.: Distributed Gaussian Processes. In: Proceedings of the 32nd International Conference on Machine Learning, PMLR 37, pp. 1481–1490 (2015)
22. Xie, A., Yin, F., Xu, Y., Ai, B., Chen, T., Cui, S.: Distributed Gaussian processes hyperparameter optimization for big data using proximal ADMM. In: IEEE Signal Processing Letters, vol. 26, no. 8, pp. 1197–1201 (2019)
23. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: theory and implementation. In: ACM Computing Surveys, vol. 51, pp. 1–35 (2018)
24. Armknecht, F., et al.: A guide to fully homomorphic encryption. In: IACR Cryptology ePrint Archive (2015)