Privacy-Preserving Support Vector Machine for Outsourcing

4 Anonymous Authors
Implementation track

Abstract

This project is a reimplementation of the first client-server privacy-persevering (pp) protocol for SVM [1]. With the recent emergence of cloud computing and data-driven machine learning methods, more and more people now aims to take advantage of these high-performance and flexible services. This leads to increasing demands of sharing private data to cloud computing providers or third-party-servers. Under this context, this project aims to fully exploited the homomorphic properties of certain encryption protocols to allow cloud servers to compute SVM without decrypting the sensitive client data. The implemented protocol not only successfully yielded the same high accuracy as the regular SVM on testing data sets but also achieved the classification with secure two-party computation.

1 Motivation

Nowadays machine learning based classifiers have gained great popularity, as they can learn from a huge set of labeled data with superior speed and accuracy compared to many traditional methods [2] [3]. However, this also means that the performance of such classifiers is highly correlated to the size and quality of the training data [4]. Many smaller organizations lack the human resource to collect huge amounts of valid data or the computational recourse to train and store large classifiers, making outsourcing classification tasks a sound choice [5]. In particular, the client will provide a private set of testing data that they want to classify. The provider will take the dataset and process through their already trained classifier and provide the results. This framework would mitigate the technical requirements for the client to take advantage of the state-of-the-art machine learning algorithms.

Cloud computing has gained its popularity as one of the most flexible outsourcing techniques. A group of powerful servers could remotely provide clients with data classifiers and computing resources. Releasing the privately owned training data samples is risky, concerning that the data in the cloud are outsourced to untrusted third-party-servers. That being said, this framework unavoidably induces problems regarding privacy and data governance. For example, if a doctor wants to use sensitive patient information to diagnose disease via cloud computing, the doctor might not be legally capable of uploading this information to an untrusted-third-party-server. On the other hand, the cloud computing provider might not wish to disclose the detailed parameters of the classifier. To solve this problem, we need a secured cloud computing channel for both the client and the provider.

2 Problem Statement

In an SVM prediction outsourcing scenario, there are two entities - the client, which provides the input data and hope to get the result, and the server, which owns the training data and performs most of the computational tasks. The objective of this project is to build a cloud-based machine learning classifier while simultaneously making sure that a) the server cannot interpret the test data or the classification result; b) the client cannot interpret the training data or classifier parameters.

To express it mathematically, let $\mathbf{x}$ denote the training set on the server, and vector $\mathbf{t}$ denote the input from the client. Recall that for an SVM with polynomial kernel, the kernel function is

$$K(\mathbf{x}_i, \mathbf{t}) = \mathbf{x}_i\mathbf{t} + 1, K^p(\mathbf{x}_i, \mathbf{t}) = (\mathbf{x}_i\mathbf{t} + 1)^p \quad (1)$$

where p is the degree. The decision function $d(\mathbf{t})$ is

$$d(\mathbf{t}) = \sum_i \alpha_i y_i K^p(\mathbf{x}_i, \mathbf{t}) + b \quad (2)$$

and the result is

$$f(\mathbf{t}) = sign(d(\mathbf{t})) \quad (3)$$

That is, during the outsourcing task, the followings are considered as privacy of the server:

- The training set $\mathbf{x}$, and its properties including mean $\mu_\mathbf{x}$ and standard deviation $\sigma_\mathbf{x}$
- SVM model parameters: α_i (and support vectors, which is considered the same category as the training set). Hyperparameters like the degree of the polynomial kernel $p$is not included
- Values that might imply the training data or model parameters: $K(\mathbf{x}_i, \mathbf{t})$, $K^p(\mathbf{x}_i, \mathbf{t})$, and $d(\mathbf{t})$.

On the client side, the privacy includes:

- The input data $\mathbf{t}$
- The classification result $f(\mathbf{t})$

Using the criteria above, whenever any sensitive information enters the other entity, it has to be either encrypted, or masked (added or multiplied by a random number r). The proposed PPSVM [1] modifies the original SVM prediction tasks (Equation 1, 2 and 3) to provide a solution to the condition above.

3 Literature Review

As cloud computing gains popularity, privacy persevering protocols that protect both training data for the server and the test data for the client have been gathering more and more attention from scholars. How to fully utilize the potential of machine learning while retaining some degree of privacy remains a challenge. Currently, scholars have proposed many different strategies, and they can generally be grouped into two categories.

The first school of thought aims to protect privacy of training data locally via pre-processing such as randomization or projection. Chen et al. [6] introduced a random rotation perturbation framework for the training process of the machine learning algorithm. Individual anonymity is protected while classification-specific information is preserved. Chaudhuri et.al. [7] and Rubinstein et.al. [8] presented a similar framework with a slightly different approach. The training data are randomly projected to another feature space, encrypting the original sensitive information while not hindering the training process. In both cases, all training data are centralized and stored in local devices. This kind of framework made it very efficient for the training and inferring process. However, locally storing distorted sensitive information still poses a huge security risk when faced with competent hackers.

The second school of thought is to completely segment the data from potential risks via encryption or partitioning. Yu et al. proposed the idea of partitioning where each party only has access to a limited portion of data [9, 10]. To train the whole model and take advantage of the entire training set, Yu et al. proposed a secure dot product method to compute the Gram matrix for all data jointly. One of them will be an “initiator” who contributes no data and performs the Gram matrix computation as well as the predictions. This method can only be applied to kernels that can be constructed from the Gram matrix. Our selected paper [1] took this idea a step further and encrypted all the communications. A more concise framework involving only the client and the server is proposed. The server only received an encrypted version of the client’s data and exploited the properties of homogeneous encryption (HE) to conduct all the computations needed for training in the encrypted domain. While computation in the encryption domain is slower than previous methods, this method greatly improves data security.

This work is often considered as one of the pioneering papers in this domain and has inspired many papers to follow [11, 12]. Li et al. [13] built upon this paper and improved a vulnerable point in the framework. Zhu et al. [14] constructed an application for online self-diagnosis using a similar framework. Bost et.al. [15] extended this encryption idea to other simple classifiers like Naïve Bayes.

Despite all these potential, there are some inherent security limitations to this type of method using additively homographic encryption. Gao et al. [12] recently identified that a multi-round execution of the proposed framework will enable the client to gather enough information to obtain the kernel values and the number of support vectors. In other words, addictively homomorphic encryption is inherently vulnerable to repetitive attacks. That being said, the privacy-preserving SVM framework by Rahulamathavan et al. [1] is still a classic framework that intervenes the encryption into machine learning and is very much worth re-implementing for this project.

4 Method

4.1 Encryption Scheme

As mentioned in Section 2, data privacy is maintained via either encryption or mask. For a value $m \in \mathbb{Z}_n$ that is considered sensitive (which is called a message), masking states that the entity can firstly generate a random number $r \in \mathbb{Z}_n$, and send the other entity $m + r$ or mr instead of m to maintain data privacy.

As for encryption, the protocol is based on **homomorphic encryption schemes**, schemes that allow calculation on encrypted data without decryption. Specifically, **Paillier cryptosystem**, one of the public-key schemes under this category, is used in the algorithm. For value $m \in \mathbb{Z}_n$, let $[\![m]\!]$ denote the encrypted m using Paillier cryptosystem.

Firstly, Paillier cryptosystem has the property of a public-key scheme: there are two set of keys called public key and private key. Any entities with the public key can encrypt m to $[\![m]\!]$, but only entities with the private key can decrypt $[\![m]\!]$ back to m.

Secondly, as a homomorphic encryption scheme, the following holds:

$$[\![m_1 + m_2]\!] = [\![m_1]\!][\![m_2]\!] \tag{4}$$

$$[\![\alpha m_1]\!] = [\![m_1]\!]^{\alpha} \tag{5}$$

where α is a scalar. These properties allow addition and multiplication (also subtraction and division, by changing the sign) can be performed on entities without a private key.

Throughout the whole task, masking is used to protect server privacy, and Paillier cryptosystem is used to protect client privacy while allowing the server to perform calculations, where the client has both the private and public keys but the server only has the public key.

4.2 Scaling and Standardization

One shortcoming of the Paillier cryptosystem, is that it only allows encryption (and decryption) from integer to integer. Therefore, to maintain the accuracy, the data need to be scaled before taken only the integer part. As at the final step of the classification in Equation 3 only matters in sign of the decision function, scaling will not have any negative influence on the result. Every step of the SVM is modified by multiplying key values by a scaling factor of γ.

Another step of the data preprocessing is standardization (original paper refers to this step as normalization). As mean and standard deviation of the training set is considered sensitive, it is performed on the server side. For the scaled client input data $\gamma\mathbf{t}$, the scaled and standardized test sample is

$$\gamma t_i = \gamma\left(\frac{\tilde{t_i} - \mu_{xi}}{\sigma_{xi}}\right), \forall i \tag{6}$$

Using properties in Equation 4 and 5, we can convert Equation 6 into

$$[\![\gamma t_i]\!] = [\![\gamma\tilde{t_i}]\!]^{\frac{1}{\sigma_{xi}}} \left[\!\!\left[\frac{\gamma\mu_{xi}}{\sigma_{xi}}\right]\!\!\right]^{-1}, \forall i \tag{7}$$

which is the formula for normalization on the encrypted scaled client input $[\![\gamma\tilde{t_i}]\!]$

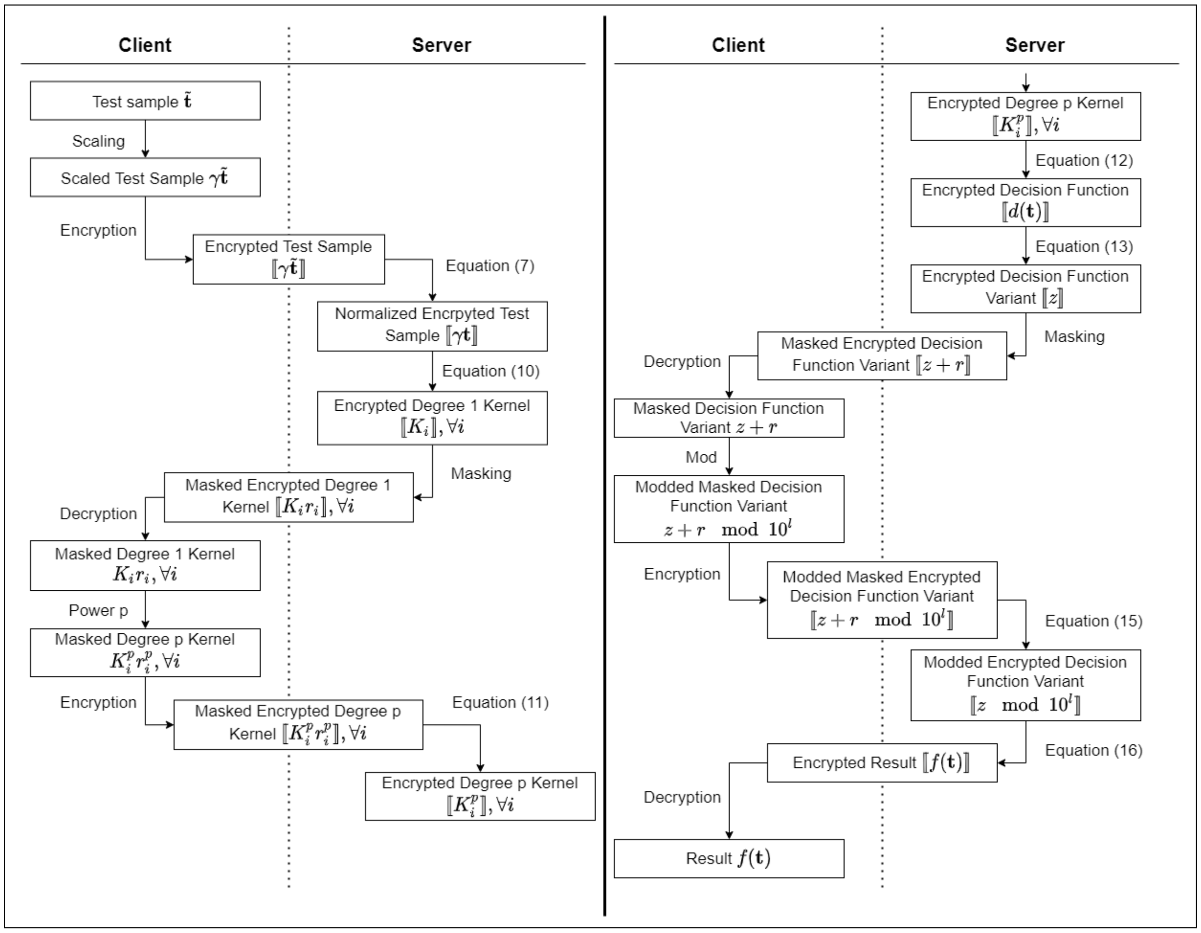
Client
Server
Test sample $\tilde{\mathbf{t}}$
Scaling
Scaled Test Sample $\gamma\tilde{\mathbf{t}}$
Encryption
Encrypted Test Sample $[\![\gamma\tilde{\mathbf{t}}]\!]$
Equation (7)
Normalized Encrpyted Test Sample $[\![\gamma\mathbf{t}]\!]$
Equation (10)
Encrypted Degree 1 Kernel $[\![K_i]\!], \forall i$
Masking
Masked Encrypted Degree 1 Kernel $[\![K_i r_i]\!], \forall i$
Decryption
Masked Degree 1 Kernel $K_i r_i, \forall i$
Power p
Masked Degree p Kernel $K_i^p r_i^p, \forall i$
Encryption
Masked Encrypted Degree p Kernel $[\![K_i^p r_i^p]\!], \forall i$
Equation (11)
Encrypted Degree p Kernel $[\![K_i^p]\!], \forall i$
Client
Server
Encrypted Degree p Kernel $[\![K_i^p]\!], \forall i$
Equation (12)
Encrypted Decision Function $[\![d(\mathbf{t})]\!]$
Equation (13)
Encrypted Decision Function Variant $[\![z]\!]$
Masking
Masked Encrypted Decision Function Variant $[\![z+r]\!]$
Decryption
Masked Decision Function Variant $z+r$
Mod
Modded Masked Decision Function Variant $z+r \mod 10^l$
Encryption
Modded Masked Encrypted Decision Function Variant $[\![z+r \mod 10^l]\!]$
Equation (15)
Modded Encrypted Decision Function Variant $[\![z \mod 10^l]\!]$
Equation (16)
Encrypted Result $[\![f(\mathbf{t})]\!]$
Decryption
Result $f(\mathbf{t})$

Figure 1: The flow diagram of the proposed algorithm.

4.3 Kernel Calculation

Kernel serves as the key part of SVM. Although there are many types of kernels, some of them might require operations that can't be performed without decryption on Paillier cryptosystem, like the Gaussian kernel which required exponential operation. Here we are focusing on the polynomial kernel $K(\mathbf{u}, \mathbf{v}) = [\langle \mathbf{u}, \mathbf{v} \rangle + 1]^p$. Consider the case $p = 1$,

$$\begin{aligned} K(\mathbf{u}, \mathbf{v}) &= [\langle \mathbf{u}, \mathbf{v} \rangle + 1] \\ &= [u_1 v_1 + u_2 v_2 + \cdots + u_n v_n + 1] \end{aligned} \tag{8}$$

As it only consists of multiplication and addition, the encrypted polynomial kernel with $p = 1$ could be expressed as

$$\begin{aligned} [\![K(\mathbf{u}, \mathbf{v})]\!] &= [\![u_1 v_1 + u_2 v_2 + \cdots + u_n v_n + 1]\!] \\ &= [\![u_1 v_1]\!][\![u_2 v_2]\!] \ldots [\![u_n v_n]\!][\![1]\!] \\ &= [\![u_1]\!]^{v_1} [\![u_2]\!]^{v_2} \ldots [\![u_n]\!]^{v_n} [\![1]\!] \end{aligned} \tag{9}$$

That is, if we have encrypted normalized test sample $[\![\gamma\mathbf{t}]\!]$, the encrypted degree 1 kernel could be calculated as

$$[\![K_i]\!] = [\![\gamma t_1]\!]^{\gamma x_{i1}} [\![\gamma t_2]\!]^{\gamma x_{i2}} \ldots [\![\gamma t_d]\!]^{\gamma x_{id}} [\![\gamma^2]\!] \tag{10}$$

For $p > 1$, however, the kernel above needs to be raised by degree p. Equation 4 and 5 cannot raise the power of a number without decrypting it. Therefore, the server will send the degree 1 kernel back to the client and let client raise the power. As in section 2 it is stated that the value of the kernel is considered sensitive, the server will mask it by multiplying by r, and remove the mask by dividing by r^p after the client sends back the result. That is,

$$[\![K_i^p]\!] = [\![K_i^p r^p]\!]^{\frac{1}{r^p}} \tag{11}$$

4.4 Decision Function and Result

With the kernel value known, the encrypted value of the decision function could be expressed as follows:

$$\begin{aligned} [\![d(\mathbf{t})]\!] &= \left[\!\!\left[\sum_i \gamma\alpha_i y_i K^p(\mathbf{x_i}, \mathbf{t}) + \gamma^{2p+1} b \right]\!\!\right] \\ &= [\![\gamma^{2p+1} b]\!] \prod_i [\![K^p(\mathbf{x_i}, \mathbf{t})]\!]^{\gamma\alpha_i y_i} \end{aligned} \tag{12}$$

However, the encryption scheme does not allow calculation of $[\![sign(d(\mathbf{t}))]\!]$ without decryption. Leaving the sign operation to the client requires server to send the value of the decision function, which is considered sensitive. To address this problem, we find a number l such that every decision function is smaller than 10^l. Then, we define

$$z = 10^l + d(\mathbf{t}), [\![z]\!] = [\![10^l]\!][\![d(\mathbf{t})]\!] \tag{13}$$

Note that the l-th digit of z from the right is 1 if $d(\mathbf{t}) \geq 0$, and 0 if $d(\mathbf{t}) < 0$. Expressing it mathematically, we have the most significant digit as

$$\tilde{z} = 10^{-l}[z - (z \mod 10^l)] \tag{14}$$

and $\tilde{z} = 1$ iff $sign(d(\mathbf{t})) \geq 0$. This way allows server to mask the decision function by adding r before sending to client for the mod operation, and the server can remove the mask by

$$[\![z \mod 10^l]\!] = [\![z + r \mod 10^l]\!][\![r \mod 10^l]\!]^{-1}[\![\lambda]\!]^{10^l} \tag{15}$$

Where $\lambda \in \{0, 1\}$ is to prevent underflow and can be judged by a secure comparison ($\lambda = 1$ if $z + r \mod 10^l < r \mod 10^l$). The most significant digit in encryption form will be

$$[\![\tilde{z}]\!] = [\![z]\!]^{10^{-l}}[\![z \mod 10^l]\!]^{-10^{-l}} \tag{16}$$

and this is the encrypted result, in form of $\{0, 1\}$.

5 Evaluation

5.1 Experimental Setup

5.1.1 Client-Server Model

In the experiment, we implemented client and server as two different Python classes. All communications between the two classes is via certain methods so that we could monitor whether the privacy is preserved. Both classes are initialized with the common information like the public key, kernel degree p. Server is given the training data, and the client is given the test data and also the private key for decryption.

The experiment starts with the client sending server the encrypted scaled test samples chosen from the test data, and the evaluation is performed on the client side after the server return the classification result.

5.1.2 Dataset

The datasets we use are popular machine learning data sets from the UCI machine learning repository [16]. The first one, in common with Ref. [1], is Wisconsin Breast Cancer (WBC). The WBC dataset has 683 instances with each instance having 9 integer attributes. 444 samples are benign (non-cancerous), and 239 samples are malignant (cancerous). The second one is Chronic Kidney Disease (CKD). The CKD dataset has 228 instances with each instance having 12 real attributes.

Among the 228 samples, 101 of them are benign, and 127 of them are malignant. The third one is Liver Disorders Data Set (BUPA) which has 345 instances, each with 7 attributes. It does not have a classification label field but a manually created selector field which does not depend on other attributes. Nevertheless, we can still utilize this data set to compare the performance of PPSVM and classical SVM. All three are medical datasets. For the purpose of consistency and SVM performance, the labels in these data sets are masked to positive (+1) labels and negative (-1) labels. They are appropriate for the project because they are binary data sets which are more straightforward to be classified by SVM, and they contain accurate and legitimate information from real, credible sources.

As it is not the focus of the project, the decision on the hyperparameters (kernel degree p and regularization constant C are decided independent of the PPSVM algorithm. The optimized parameters for the three datasets are:

- WBC: $p = 2$, $C = 0.005$,
- CKD: $p = 3$, $C = 0.012$,
- BUPA: $p = 2$, $C = 0.18$.

For the WBC dataset, there are discrepancies between the data from Ref. [16] and the description of it in Ref. [1]. For example, Ref. [1] reported 237 malignant samples whereas we have 239. Ref. [1] also noted that the number of support vectors is around 10, without specifying parameter C. However, even with $C > 10^4$ and normalized data, we get > 50 support vectors. As a result, the accuracies we get from unencrypted SVM are also different from those in Ref. [1]. Since we use the same dataset and training process for PPSVM and SVM and focus on the difference between the two, this does not undermine our main results.

5.1.3 Performance Measurements

The performance measurements focus on the comparison between our PPSVM accuracy and regular SVM accuracy. Thus, we keep the parameters of them the same (namely, α and support vectors). By design, the difference in accuracy between PPSVM and regular SVM only comes from rounding to integer before encryption, as discussed in Section 4.2. A large scaling factor γ reduces this effect and is expected to improve accuracy. The dependence of accuracy on γ is to be studied, in order to find out a threshold of γ that will give us negligible accuracy loss. We evaluate per-class and overall accuracies for both SVMs and compare between them.

Besides, we are also interested in the time cost of the algorithm comparing to the original paper, as computational time might be negatively influenced by the encrypted scheme. To ensure the algorithm is practical, it has to stay within reasonable computational time for reasonable hyperparameters and model complexity. In this regard, we also monitor the computational time with different datasets and hyperparameters. The sum of the computational time for each critical step, including all encrypted arithmetic operations, is recorded with Python's `time` module.

We employ the leave-one-out testing approach in accordance with Ref. [1]. It has the advantage of maximizing the usage of a dataset in training while also avoiding overfitting. For a single dataset of N samples, we fit N models, each with one different sample as the testing set and with the remaining $N - 1$ samples as the training set. The N models will then make predictions for all N samples and the accuracy is defined as ratio between the number of correct predictions and sample size N. Per-class accuracies can be defined similarly as the number of correct predictions within samples that have a certain label divided by the number of samples with that label.

5.2 Results

5.2.1 Classification Accuracy vs. Scaling Factor

The evaluation results for WBC, CKD, and BUPA datasets are presented in Fig. 2 and 3. In the left panel of each figure we show the per-class and overall accuracies of PPSVM and regular SVM as a function of the scaling factor γ ranging from 10^0 to 10^6. For all three datasets, PPSVM have large differences in accuracy compared to regular unencrypted SVM when γ is smaller than 10^3. However, the gap becomes smaller when $\gamma = 10^3$, where the difference between PPSVM and SVM is consistently smaller than 3% for all three datasets. Note that starting from 10^4, PPSVM and SVM make exactly same predictions. This is because the data have finite significant figures and multiplied by the scaling factor they become integers and are thus not affected by the rounding process. For the WBC dataset, we reach equal accuracies for PPSVM and SVM at the same scaling factor ($\gamma = 10^4$) as Ref. [1].

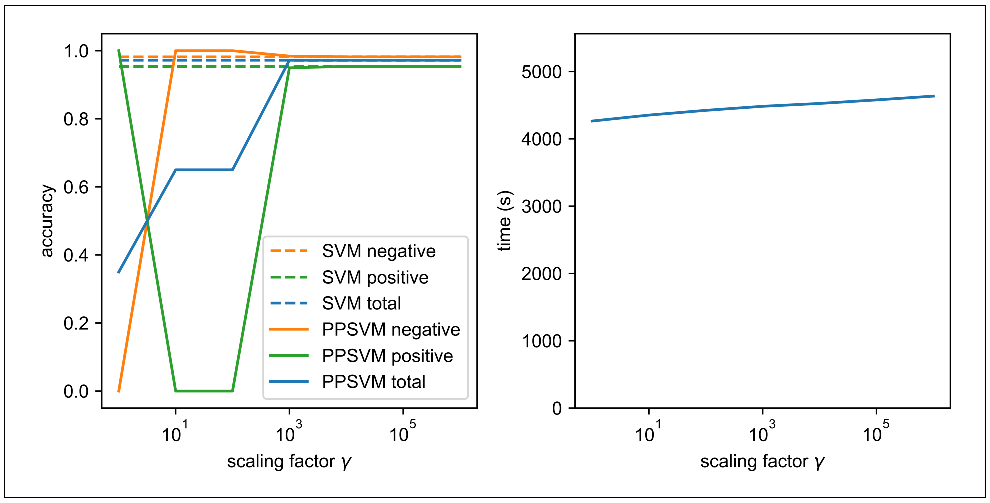

Figure 2: Evaluation results on WBC dataset.

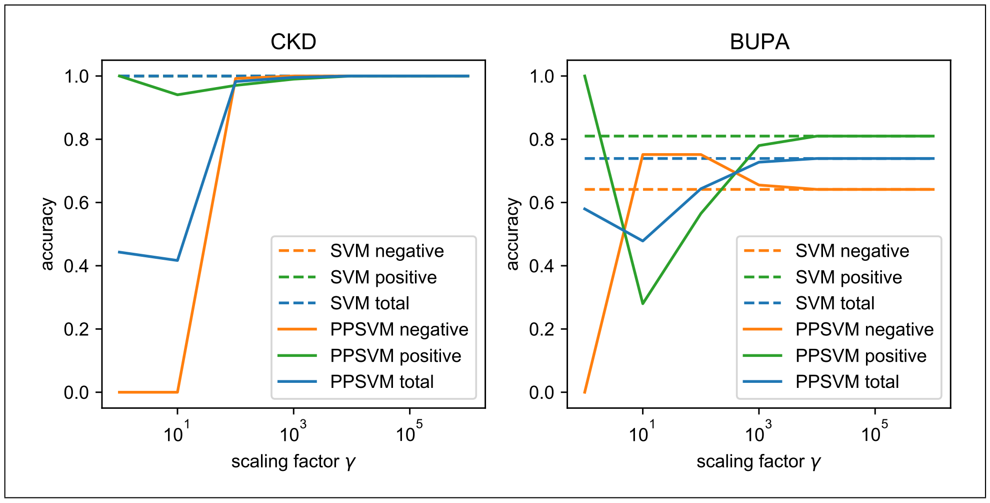

Figure 3: Evaluation results on CKD and BUPA datasets.

5.2.2 Computational Time

The total computational time of a leave-one-out validation cycle for the entire WBC dataset is over 1 hour, and given the size of the dataset this is beyond the practical range, and is different from Ref. [1] with a runtime of 7.71 seconds. The other two datasets also both have impractical runtimes over 30 minutes.

We conclude that the expensive time cost is caused by the external library we use for Paillier cryptosystem, which does not support direct encryption, decryption, or calculation on NumPy arrays. In this case, the computational time will mostly depend on the size of the matrice in the whole algorithm, which the largest is generally the kernel matrix with size of (n_{sv}, n_{input}) where n_{sv} is the number of support vectors, and n_{input} is the number of input samples to be classified ($n_{input} = 1$ in our leave-one-out validation scheme). This is supported by further experiments with result shown in Fig. 4, where the prediction time grows significantly with the number of support vectors, all with degree 2 kernels.

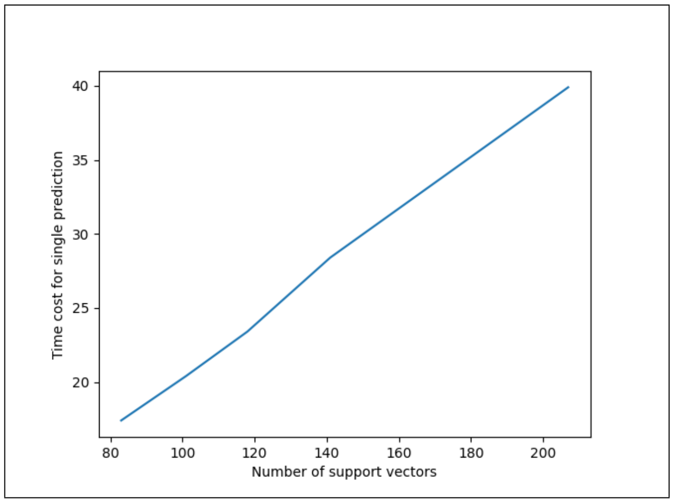

Figure 4: Computational time on single prediction vs Number of support vectors on WBC dataset.

As shown on the right side of Fig. 2, the runtime is also correlated with the scaling factor. Experiments on the other two datasets also showed similar behaviors. The reason it is correlated with scaling factor is not clear, probably attributes to the nature of the package we are using, especially to how large integers are handled. The time cost can be optimized by implementing the cryptosystem from scratch as the original paper did, but due to the resource limitation we decided not to pursue it.

6 Conclusion

In this project, we implemented the proposed PPSVM algorithm and confirmed the privacy-preserving nature of the protocol. We found out the accuracy is fairly high, and could easily reach the level of the classical SVM with proper hyperparameters.

However, we do believe that there are many shortcomings to convert the theory into practice. One of the biggest drawbacks is overflow problem. Although scaling does not influence the output of the SVM, the decision function will generally be larger than γ^{2p+1}, with p being the degree of the polynomial kernel. From experiments we observed a scale factor of 10^4 is practically reasonable, and even with a degree 1 kernel, the decision function might be larger than 10^{12}, which is already larger than a 32-bit int. Another problem is from the external libraries for the cryptosystem. There are hardly any available Paillier system libraries that could directly support matrix operations (like direct encryption, decryption and calculation on NumPy in Python). Use of external libraries would significantly increase the time cost, but implementing the cryptosystem from scratch would also requires proper support on matrice.

Overall the project is a success, with a finished implementation of the proposed algorithm and similar results as ones from original paper except the time cost. We also discovered some design redundancy in the original paper. For example, the last step in the algorithm uses a mod operation serving as the sign operation, which we found superfluous, as it still depends on a secure comparison to decide the underflow factor λ in Equation 15. With a secure comparison required, it might be just more convenient to conduct a secure comparison between the decision function $d(\mathbf{t})$ and 0.

On our end, there are a few minor details that we can further work on given more time. For example, our choice of cryptosystem packages led us to the current unsatisfying runtime of the program. Moreover, when writing the proposal, we did not notice that training α is the same as the classical SVM. As the project went on, we decided to use external packages (sklearn) for those tasks so that we could focus on the parts different from the classical SVM.

References

[1] Yogachandran Rahulamathavan, Raphael C-W Phan, Suresh Veluru, Kanapathippillai Cumanan, and Muttukrishnan Rajarajan. Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud. *IEEE Transactions on Dependable and Secure Computing*, 11(5):467–479, 2013.

[2] Timo M Deist, Frank JWM Dankers, Gilmer Valdes, Robin Wijsman, I-Chow Hsu, Cary Oberije, Tim Lustberg, Johan van Soest, Frank Hoebers, Arthur Jochems, et al. Machine learning algorithms for outcome prediction in (chemo) radiotherapy: An empirical comparison of classifiers. *Medical physics*, 45(7):3449–3459, 2018.

[3] Michael H Goldbaum, Pamela A Sample, Kwokleung Chan, Julia Williams, Te-Won Lee, Eytan Blumenthal, Christopher A Girkin, Linda M Zangwill, Christopher Bowd, Terrence Sejnowski, et al. Comparing machine learning classifiers for diagnosing glaucoma from standard automated perimetry. *Investigative ophthalmology & visual science*, 43(1):162–169, 2002.

[4] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[5] Smitha Sundareswaran, Anna Squicciarini, and Dan Lin. Ensuring distributed accountability for data sharing in the cloud. *IEEE transactions on dependable and secure computing*, 9(4):556–568, 2012.

[6] Keke Chen and Ling Liu. Privacy preserving data classification with rotation perturbation. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–, November 2005.

[7] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research*, 12(29):1069–1109, 2011.

[8] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a Large Function Space: Privacy-Preserving Mechanisms for SVM Learning. *Journal of Privacy and Confidentiality*, 4(1), July 2012.

[9] Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 603–610, New York, NY, USA, April 2006. Association for Computing Machinery.

[10] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-Preserving SVM Classification on Vertically Partitioned Data. In *Advances in Knowledge Discovery and Data Mining*, pages 647–656. Springer, Berlin, Heidelberg, April 2006.

[11] Jinwen Liang, Zheng Qin, Jianbing Ni, Xiaodong Lin, and Xuemin Sherman Shen. Efficient and privacy-preserving outsourced svm classification in public cloud. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[12] Chongzhi Gao, Jin Li, Shibing Xia, Kim-Kwang Raymond Choo, Wenjing Lou, and Changyu Dong. Mas-encryption and its applications in privacy-preserving classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[13] X. Li, Y. Zhu, J. Wang, Z. Liu, Y. Liu, and M. Zhang. On the Soundness and Security of Privacy-Preserving SVM for Outsourcing Data Classification. *IEEE Transactions on Dependable and Secure Computing*, 15(5):906–912, September 2018.

[14] Hui Zhu, Xiaoxia Liu, Rongxing Lu, and Hui Li. Efficient and privacy-preserving online medical prediagnosis framework using nonlinear svm. *IEEE journal of biomedical and health informatics*, 21(3):838–850, 2016.

[15] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[16] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.