A Protocol for Privacy Preserving Neural Network Learning on Horizontally Partitioned Data

Nico Schlitter

Faculty of Computer Science
Otto-von-Guericke-University Magdeburg, Germany
nico.schlitter@iti.cs.uni-magdeburg.de

Abstract. This article presents a new approach for privacy preserving neural network training. Several studies have been devoted to privacy preserving supervised model learning, but little work has been done to extend neural network learning with a privacy preserving protocol. Neural networks are popular for many applications, among else those calling for a robust learning algorithm. In this study, we elaborate on privacy preserving classification as well as regression with neural networks on horizontally partitioned data. We consider a scenario of more than two parties that are semi-honest but curious. We extend the neural network classification algorithm presented by Rumelhart et al. with protocols for secure sum and secure matrix addition. The extended algorithm does not fully guarantee privacy in the sense defined by Goldreich, but we show that the information revealed is not associated to a specific party and could also be derived by juxtaposing the local data and the final model.

1 Introduction

In recent years a new trend in medical diagnosis came up: The so-called *computer-aided diagnosis* (CAD) procedure automatically analyzes physical examination results and provides, based on the results, diagnosis support information for medical personnel. In particular in the area of medical image recognition promising results were achieved [1]. CAD systems are widely used to recognize pathological regions in medical images. For this, neural networks are often used to extract the relevant patterns from images. The application of neural networks for this task is mainly motivated by their robustness to noisy data and their ability to determine general patterns by data generalization. Furthermore, neural networks support incremental learning, i.e., it is possible to update a trained network by presenting new training data to the network. The training data is a set of vectors generated by applying feature extraction methods on medical images. Each medical image is used to generate a feature vector that is then classified during the application phase by the network as *healthy* or *pathological*. For this, the neural network builds an internal model that maps a feature vector to one of the two classes. The trained network is applied in CAD systems in order to classify new medical images.

Data sources are image forming technologies, such as Mammography [2], Magnetic Resonance Imaging [3, 4] and Ultrasonography [5]. The mentioned techniques are used to support breast cancer diagnosis. Phonovibrography, a method for processing and visualizing the vocal fold vibrations was recently introduced by Lohscheller et al. [6]. Based on this approach, a method for automatically extracting numerical features describing the spatio-temporal and symmetrical behavior of the vocal folds was proposed in [7]. For CAD purposes, the extracted features could be used to train a classifier in order to distinguish between healthy and pathological vocal fold vibration patterns [8]. Chest Radiography [9], another image forming technique, provides information for lung cancer recognition via a neural network classifier.

The classification quality of neural network depends, amongst others, on the number of training data. Therefore, the classification performance can be increased by using images from more than one health care institution. However, in this case, privacy issues come up because medical images are considered as sensitive data. As a consequence, conventional methods for knowledge discovery from data are not appropriate; in this case, their use is even restricted by privacy protection laws, such as the Health Insurance Portability and Accountability Act (HIPAA). Privacy preserving data mining methods offer the chance to build models and extract patterns without disclosing private data.

Privacy preservation methods for knowledge discovery can be categorized into two groups, data perturbation methods and cryptographic methods. Methods of the first group use data distortion, such as adding uniform noise, with the purpose of "hidding" private data, or, more formally, of guaranteeing *k-anonymity*. Methods of the second group are used for collaborative model learning: $P \geq 2$ parties contribute their data for the learning of a shared model according to protocols [10, 11] that prevent the disclosure of the contributed data.

Our work belongs to the second group of methods. Cryptographic methods have been proposed for supervised learning, among else with decision trees, support vector machines and Bayesian networks, for unsupervised model learning, e.g., with K-means and for association rule discovery. We focus on supervised learning and discuss privacy preservation for neural network training. We assume the scenario of more than two semi-honest parties. The data contributed among these parties are *horizontally partitioned*, in the sense that the shared model is built upon the union of the contributed datasets.

The rest of the paper is organized as follows. The next section gives a brief overview on privacy preserving data mining in general and on the rarely studied issue of privacy preserving neural network learning. In Section 3, we present a widely used neural network learning method - the backpropagation learning. The extension of this method to provide for privacy preservation is discussed in Section 4. Subsequently, we discuss the limits of the protocol and the implications for applications. Section 5 concludes with a summary and some directions for future work.

2 Related work

A considerable amount of methods for privacy preservation in data mining use cryptography techniques from the Secure Multi-party Computation (SMC) area based on the seminal works by Yao [10] and Goldreich et al. [11]. In [12–14] several relevant operations for SMC were defined. These operations are applied, amongst others, in the following data mining algorithms to ensure privacy preservation: Decision tree induction was enhanced for vertically [15, 16] and horizontally [17] partitioned data to generate decision trees without data disclosure. Privacy preserving association rule mining [18] was proposed as well as clustering methods [19]. A summary of data mining applications and their privacy preserving solutions is given by Vaidya et al. [20]. However, in the area of neural networks, the aspect of privacy preservation is mostly disregarded.

Wan et al. present a generic formulation for secure computation of gradient descent methods [21]. The authors discuss a multi-party-protocol for vertically partitioned data that can be used to train a neural network. To ensure privacy, the target function is defined as a composition of two functions. Thus, the weights can be adapted locally. A second protocol for a secure summation of two scalar products is also suggested as a part of the overall process.

A privacy preserving version of self organizing maps (SOM) is presented by Han et al. [22]. SOMs belong to the class of unsupervised learning techniques and are applied, e.g., for dimension reduction. The authors present a two-party protocol to adapt the network weights iteratively for vertically partitioned data.

Barni et al. address in [23] a two-party privacy preserving protocol for neural network based computation. In their setting, the first party owns the confidential data and the second party owns the confidential model that is applied on the first party's data. Both, the data and the network model, are kept private. The approach does not consider how the used network model is actually trained but assumes that it already exists.

In the following, we present our approach which extends the presented works by proposing a multi-party protocol for privacy-preserving neural network learning on horizontally partitioned data.

3 Backpropagation algorithm for neural network learning

Neural Networks are nature inspired computation models that are widely used for regression and classification tasks. A neural network consists of nodes and weighted edges. In general, we distinguish feed forward and recurrent neural networks. In a feed forward network, the information is transmitted only in one direction, from the input nodes, through the hidden nodes (if any) to the output nodes. A feed forward network has therefore no cycles or loops. In recurrent neural networks, connections between nodes form a directed cycle. Therefore, information might also be transmitted backwards. An example of a feed forward network is shown in Fig. 1. Each node, also called neuron, produces an output by applying the internal activation function A on the weighted neuron inputs. The

output is weighted by the edge weight and the updated value is forwarded to the next layer. By this, an input vector traverses from the input layer through the network and produces an output in the output layer. In other words, the function performed by the network is determined by the nodes' activation function and the internal network weights.

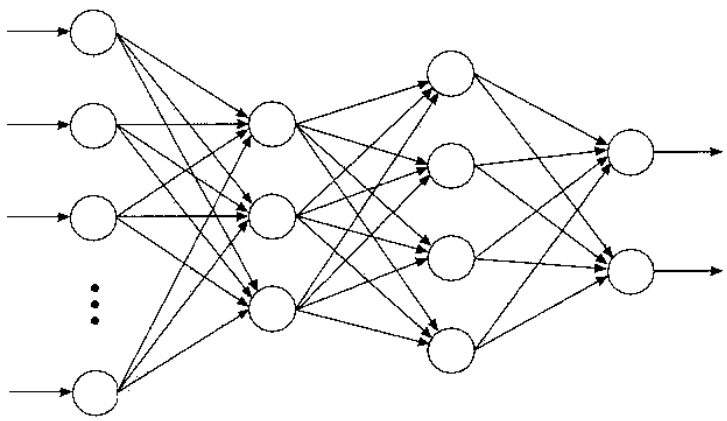

Fig. 1. Feed forward network

The aim of neural network training is to approximate an application-dependent objective function. A popular method for feed forward neural network learning is the backpropagation algorithm proposed by Rumelhart et al. [24] which we present in the following.

We assume a data set $D = \{x_n, y_n \mid n = 1, 2, \ldots, N\}$ where each vector $x_n = \{x_n(1), x_n(2), \ldots, x_n(q)\}$ represents the n-th training vector with dimension q. If the vector $y_n = \{y_n(1), \ldots, y_n(m)\}$ corresponds to the m-dimensional target vector, we approximate the target function $F : x_n \mapsto y_n$ that maps x_n to y_n. The function approximation is a complex task and usually a deviation between the approximate f and the original F remains. In order to measure this difference the mean squared error (MSE) is defined in Equation 1. Here, the vector $\hat{y}_n = f(x_n)$ refers to the m-dimensional output of the neural network while x_n is the vector that is used as the network input.

$$MSE = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{m} \left(\hat{y}_n(i) - y_n(i) \right)^2 \tag{1}$$

The objective of neural network training is to adapt the internal network weights in order to minimize the MSE. Usually the backpropagation algorithm [24] is iteratively used to reach this goal. At first, the network weights are randomly initialized. In the next step, the weights are iteratively adapted in order to reach

a local and in the best case a global optimum. In each round t of the training process the algorithm adapts the weight W_{ij}^l between the i-th neuron in layer l and the j-th neuron in layer $l+1$.

$$W_{ij}^l(t+1) = W_{ij}^l(t) + \Delta W_{ij}^l(t) \tag{2}$$

In the online-mode the weight adjustment ΔW_{ij}^l is performed after one learning iteration. For this, after randomly choosing one training vector x_n, the network is learned and the error is calculated. Afterwards, the error is propagated back to the input layer and the weights are recalculated accordingly. However, here we refer to the batch-mode where a weight adjustment is done after all N training vectors have been applied to learn the network. $W_{ij}^l(t)$ is then the sum of N changes $\Delta W_{ij}^l(t,n)$ caused by the n-th data record.

$$\Delta W_{ij}^l(t) = \sum_{n=1}^{N} \Delta W_{ij}^l(t,n) \tag{3}$$

The calculation of the weight adjustment $\Delta W_{ij}^l(t,n)$ in round t is shown in Equation 4 where $N_i^l(t,n)$ refers to the output of the i-th neuron in layer l while x_n is the vector that is used as input for the whole network. The parameter γ controls the size of the adaptation steps on the error surface and is called learning rate.

$$\Delta W_{ij}^l(t,n) = -\gamma(t) \cdot N_i^l(t,n) \cdot \delta_j^{l+1}(t,n) \tag{4}$$

The error $\delta_j^{l+1}(t,n)$ is the fraction of the network error that is caused by the j-th neuron in layer $l+1$. The computation of the error depends on the location of the respective neuron in the network and is shown in Equation 5.

$$\delta_i^l(t,n) = \begin{cases} A_i'^l\Big(\hat{y}_n(i)\Big) \cdot \Big(\hat{y}_n(i) - y_n(i)\Big) & \text{if } l = l_{max} \\ A_i'^l\Big(N_i^l(t)\Big) \cdot \sum_j W_{ij}^l(t) \cdot \delta_j^{l+1} & \text{if } l < l_{max} \end{cases} \tag{5}$$

In case the neuron is located in the output layer l_{max} the error signal is calculated as the product of the first derivative of the neuron's activation function A and the current network error. The error signal of the inner neurons is computed recursively as the product of two factors. The first factor is the output of the first derivative of the activation function applied to the neuron's output. The second factor is the weighted sum of the error signals that are propagated through the network.

The iterative weight adaption described in Equation 2 is stopped when the maximum number of training rounds is reached or the MSE (cf. Equation 1) falls below a previously defined threshold. A more detailed description of the back-propagation algorithm and the proof of convergence can be found in [24].

4 A protocol for privacy preserving backpropagation

In this section we propose a protocol for privacy preserving neural network training. We present a formalization of the given problem and introduce our training approach based on SMC methods - in particular the Secure Sum Protocol.

4.1 Problem setting

We assume a scenario with P parties that want to collaborate for a common data analysis task using a neural network. The data is horizontally partitioned, meaning that all parties share the same data schema and each party holds a subset of the data containing several records of the total data set D. Analogously to the definitions in Section 3, the data set of the p-th party is specified as $D_p = \{x_n^p, y_n^p\}$ and contains $N_p = |D_p|$ records. As a consequence the data set $D = \bigcup_{p=1}^{P} D_p$ only exists in virtual form: it is distributed over P parties and is never merged because of privacy concerns. Additionally we suppose a semi-honest behavior of the participating parties. Thus, we assume, that all parties adhere to the given protocol. However, each party is curious and attempts to derive additional information about the other parties. We are also aware of colluding parties that work together against one or more participants in order to obtain private information about other parties. Consequently the proposed protocol needs to deal with the described situation and has to protect the participants' privacy. The common objective of all parties is to train the neural network in order to minimize the overall MSE:

$$MSE = \frac{1}{\sum_{p=1}^{P} N_p} \sum_{p=1}^{P} \sum_{n=1}^{N_p} \sum_{i=1}^{m} \left(\hat{y}_n^p(i) - y_n^p(i) \right)^2 \tag{6}$$

In the next Section, we present our proposed protocol which ensures, that the underlying private datasets D_p and related knowledge are kept private.

4.2 The protocol execution

In the following, we describe our proposed protocol for privacy preserving neural network training. The procedure is shown in Algorithm 1. Basically we extend Rumelhart's original algorithm [24] to the multi-party case. Thus, we adapt the formula for the weight adjustment (cf. Equation 3) in order to handle the weight adaptations that are determined by different parties. In Equations 7 to 9 we show, that Equation 3 is equivalent to 9.

$$\Delta W_{ij}^l(t) = \sum\nolimits_{n=1}^{N} \Delta W_{ij}^l(t,n) \qquad for\ N = \sum N_p \tag{7}$$

$$= \sum\nolimits_{p=1}^{P} \sum\nolimits_{n=1}^{N_p} \Delta W_{ij}^l(t,n) \tag{8}$$

$$= \sum\nolimits_{p=1}^{P} \Delta W_{ij}^{lp}(t) \tag{9}$$

Each party holds a subset of the complete data set D and computes the weight adjustments based on its own local data records. Equations 7 to 9 show that the sum of these weight changes is equal to the sum of the subtotals of weight changes as a consequence of the associative law. Hence, the neural network trained by our protocol is equal to a network generated by Rumelhart's algorithm.

Algorithm 1: Privacy preserving backpropagation network training

input : distributed data set $D = \bigcup_{p=1}^{P} D_p$
output: weight matrix W_{ij}^{l}

1 All parties agree on network topology and type of activation function;
`Initiate the global network`
2 **foreach** *party* P **do**
  `Randomly choosing of local weight matrix`
3 $W_{ij}^{lp} = rand();$
`Create global weight matrix by using secure matrix addition`
4 $W_{ij}^{l} = \sum_{p=1}^{P} W_{ij}^{lp};$
`Start network training`
5 **repeat**
6 **foreach** *party* p **do**
7 `Copy global to local weight matrix`
  $W_{ij}^{lp} = W_{ij}^{l};$
8 `Compute local weight changes`
  $\Delta W_{ij}^{lp} = \sum_{n=1}^{N_p} -\gamma \cdot N_i^l(n) \cdot \delta_j^{l+1}(n);$
9 `Compute local squared error`
  $MSE^p = \sum_{n=1}^{N_p} \sum_{i=1}^{m} \left(\hat{y}_n^p(i) - y_n^p(i) \right)^2;$
10 `Compute global weight changes by using secure matrix addition`
  $\Delta W_{ij}^{l} = \sum_{p=1}^{P} \Delta W_{ij}^{lp};$
11 `Adapt global weight matrix`
  $W_{ij}^{l} = W_{ij}^{l} + \Delta W_{ij}^{l};$
12 `Compute MSE by using secure sum protocol`
  $MSE = \frac{1}{\sum_{p=1}^{P} N_p} \sum_{p=1}^{P} MSE^p;$
13 **until** *MSE is small enough* ;
14 **return** W_{ij}^{l};

The execution of the proposed protocol works as follows. According to the original algorithm the parties randomly initiate the network weights (cf. line 2 in Algorithm 1). For this, the parties have to agree upon a network topology by defining the number of network layers, the number of neurons per layer and the applied activation functions (cf. line 1 in Algorithm 1). In the next step, each party p creates a local initial weight matrix $W_{ij}^{lp}(0)$ by choosing a random number for each edge weight (cf. line 3 in Algorithm 1). Thus, each party builds

a local network and the global weight matrix is computed as the sum of all local weight matrices (cf. line 4 in Algorithm 1):

$$W_{ij}^{l}(0) = \sum_{p=1}^{P} W_{ij}^{lp}(0) \tag{10}$$

In the learning step, the weights are iteratively adapted in order to minimize the total MSE (cf. line 6-9 in Algorithm 1). For this, each party presents its training vectors x_n^p to the network's input layer. The vector is processed through the different network layers and at the network's output layer, the error between the actual and the desired output is measured. This error is afterwards minimized by back propagating it through the network and by adjusting the weight matrices according to Equation 3. By adding up the local weight changes, the global changes are computed as shown in Equation 9. The resulting weight change matrix $\Delta W_{ij}^{l}(t)$ is published to all participating parties (cf. line 10 in Algorithm 1). Thus, the new weight matrix of the network is adapted by each party locally (cf. line 11 in Algorithm 1):

$$W_{ij}^{l}(t+1) = W_{ij}^{lp}(t+1) = W_{ij}^{l}(t) + \Delta W_{ij}^{l}(t) \tag{11}$$

The weight adaption is repeated until the total MSE reaches a minimum or a convergence criterion is met. In order to calculate this total MSE Equation 6 is used (cf. line 12 in Algorithm 1).

To ensure privacy, the sums in Equations 10 and 9 are computed by using the *secure sum* (cf. line 12 in Algorithm 1) and the *secure matrix addition* protocol (cf. line 4 and 10 in Algorithm 1) which we discuss in the following.

Secure sum protocol The secure sum protocol as presented in [13] allows the secure summation of distributed summands. During the protocol's execution only the final sum but not the single summands are published. Assuming a multi-party scenario with $P > 2$ parties and respectively P private values $v_1, \ldots, v_P$ the protocol's objective is to compute the sum $S = \sum_{i=1}^{P} v_i$ that is known to be in the interval $[0, S_{max}]$. In the following, we briefly sketch the procedure.

Party 1 randomly chooses a uniformly distributed value R and sends the sum $R + v_1 \pmod{S_{max}}$ to party 2. The combination of the uniformly distributed value R and the modulo operation ensures, that party 2 learns nothing about v_1. Party 2 adds its own private value v_2 and sends the result to the next party. In general each party p receives $R + \sum_{i=1}^{p-1} v_i \pmod{S_{max}}$ from its predecessor and sends $R + \sum_{i=1}^{p} v_i \pmod{S_{max}}$ to its successor. The last party sends the masked sum back to party 1 which is able to compute the correct sum by subtracting its randomly chosen value R:

$$S = R + \sum_{i=1}^{P} v_i \quad - R \pmod{S_{max}} = \sum_{i=1}^{P} v_i \tag{12}$$

The secure sum protocol is vulnerable in case two or more parties work together (colluders). For example, if parties $p-1$ and $p+1$ collaborate, they are able to

calculate $p's$ private value v_p:

$$v_p = R + \sum_{i=1}^{p-1} v_i - \sum_{i=1}^{p} v_i \pmod{S_{max}} \tag{13}$$

This problem can be solved by splitting the private value of each party into different shares. The sum of the shares of all parties is then computed separately. However, the order of the participating parties is permuted for each share in order to guarantee different predecessor and successor for each iteration. To compute the private value v_p all predecessors and successors of party p have to collude. The protocol's security depends on the number of iterations and the number of colluding parties. The more iterations are performed, the more parties might collude without risking privacy breaches. Thus, the fraction of colluders to semi-honest parties and the number of iterations determine the *security level* of the protocol.

Secure matrix addition Based on the secure sum protocol Yu and Vaidya propose a protocol for secure matrix addition [25]. For all components of the P private matrices $W_1, \ldots, W_P$ the secure sum protocol is simultaneously executed. By this, the matrix sum $\sum_{p=1}^{P} W_p$ is calculated in one step instead of an iterative computation for each matrix component. Since the protocol also uses the secure sum protocol, the same security restrictions apply, i.e., the privacy also depends on the fraction of colluders.

4.3 Security analysis

The security of a protocol for Secure Multi-party Computation with respect to information disclosure is usually evaluated with a procedure called *privacy by simulation*: A protocol is considered secure if the view of each party during the execution can be simulated by using only its local input and the final output of the protocol. This is equivalent to saying that a party learns nothing beyond what it knew originally and what is inherent in the final result. Details on the notion of security in SMC can be found in [26].

Our algorithm does not fulfill the security requirements of SMC as presented above, because each party obtains more data during the execution of protocol than is inherent in the final model. However, the parties still do not obtain information about any *specific other party*. In particular:

In line 10 of Algorithm 1 the global weight change matrix $\Delta W(t)$ is computed by using the secure matrix addition protocol. The new weight matrix $W(t+1)$ is calculated (cf. line 11) and published to all parties (cf. line 7). The new weight matrix is used for next local network training iteration. By subtracting the new weight matrix $W(t+1)$ and the local weight changes $\Delta W^p(t)$ from the old weight matrix $W(t)$ each party is able to deduce the sum of weight changes $\Delta W^{\overline{p}}(t)$ proposed by all other participating parties:

$$\Delta W^{\overline{p}}(t) = W(t+1) - W(t) - \Delta W^p(t) \tag{14}$$

$$= \Delta W(t) - \Delta W^p(t) \tag{15}$$

The same argumentation holds for the calculation of the number of records and the MSE. Both values can be inferred by using the secure sums computed in line 12. The number of records hold by the other parties can be determined as

$$N_{\overline{p}} = \left(\sum_{i=1}^{P} N_i \right) - N_p \tag{16}$$

The MSE caused by the other parties' data can be calculated as

$$MSE^{\overline{p}} = \frac{1}{N_{\overline{p}}} \left[\left(\sum_{i=1}^{P} MSE^i \right) - MSE^p \right] \tag{17}$$

We would like to emphasize, that $\Delta W^{\overline{p}}(t)$ contains the weight changes of *all other* parties excluding party p, $N_{\overline{p}}$ refers to the number of data records that are not held by p and $MSE^{\overline{p}}$ represents the MSE of the other parties. Therefore, a single attacker is only able to obtain group related information and no information about individual parties. Using the inferred group related information, an attacker is able to detect differences between data set characteristics in its own data set D_p and the remaining data set $\bigcup_{i \neq p} D_i$. For example in each iteration, the attacker can compare its own MSE^p with the $MSE^{\overline{p}}$ of the other parties. If both MSEs are similar, the attacker can assume that the data set characteristics are similar as well. Otherwise, in case the MSEs differ considerably, it can be speculated, that the characteristics vary as well. Analogously, an attacker might use the weight change matrix in order to compare its own data set characteristics with the other party's. However, an attacker is not able to compare its data set to another *specific party* or even to reconstruct other party's data records.

If parties collude, the security depends on the security of the secure sum protocol. Even if the highest security level for the secure sum protocol is applied, the private information of a single party can be revealed if all $P - 1$ parties collude. Therefore, the security of our proposed protocol is ensured if all parties are semi-honest and do not collude. Furthermore, the protocol is secure if the number of colluders is sufficiently small with respect to the security level of the secure sum protocol.

Even though, our protocol does not fulfill the security requirements stated by the SMC community, it fulfills a more general privacy definition which is nevertheless relevant for practical applications. Vaidya et al. define two requirements for privacy preserving data mining techniques: a) Any information disclosed cannot be traced to an individual or b) does not constitute an intrusion. As discussed above, during the execution of our protocol, only group-related information is disclosed and the data revealed in each iteration does not contain any information about individual parties.

Actually, the information that can be inferred from the data which is disclosed after each iteration has no more value to any party than the information that is included in the final model. Following this perspective, one could argue, that the protocol does also fulfill SMC privacy requirements since only data but no further information than that provided by the final model is revealed during the execution of the protocol.

5 Conclusion

In this paper we proposed a privacy preserving protocol for neural network training on horizontally partitioned data. We used the secure sum and secure matrix addition protocol in order to extend Rumelhart's algorithm to a multi-party scenario. By this, the weight adaptations of the participating parties are iteratively included in the global neural network weight matrix. The protocol is not secure in the strict sense of privacy-by-simulation but the disclosed information is not associated to a single party (for $P > 2$ parties). Rather, a party can only compare the characteristic of the own local data set to the characteristic of the other parties' data sets - an operation that can be performed also by juxtaposing the local data to the final model. On this account, we consider the proposed protocol as unsecure in the strict SMC sense but sufficiently secure for practical applications. Designing a SMC-secure privacy preserving protocol for neural network learning is still an open task for future work. We consider it also important to identify scenarios, where a SMC-secure protocol is indispensable, and study the interplay of data skew and SMC-security.

References

1. Giger, M.L.: Computer-aided diagnosis of breast lesions in medical images. Computing in Science and Engineering **02**(5) (2000) 39–45
2. Lasztovicza, L., Pataki, B., Szekely, N., Toth, N.: Neural network based microcalcification detection in a mammographic cad system. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings of the Second IEEE International Workshop on (Sept. 2003) 319–323
3. Arbach, L., Stolpen, A., Reinhardt, J.M.: Classification of breast mri lesions using a backpropagation neural network (bnn). In: ISBI, IEEE (2004) 253–256
4. Meinel, L.A., Stolpen, A.H., Berbaum, K.S., Fajardo, L.L., Reinhardt, J.M.: Breast mri lesion classification: Improved performance of human readers with a backpropagation neural network computer-aided diagnosis (cad) system. Journal of Magnetic Resonance Imaging **25** (2006) 89 – 95
5. Chen, D.R., Chang, R.F., Huang, Y.L.: Computer-aided Diagnosis Applied to US of Solid Breast Nodules by Using Neural Networks. Radiology **213**(2) (1999) 407–412
6. Lohscheller, J., Eysholdt, U., Toy, H., Dllinger, M.: Phonovibrography: Mapping high-speed movies of vocal fold vibrations into 2-d diagrams for visualizing and analyzing the underlying laryngeal dynamics. Medical Imaging, IEEE Transactions on **27**(3) (March 2008) 300–309
7. Doellinger, M., Lohscheller, J., McWorther, A., Voigt, D., Kunduk, M.: Support vector machine classification of different phonatory tasks revealed by phonovibrogram analysis and acoustic perturbation measurements. The Journal of the Acoustical Society of America (2008)
8. Lohscheller, J., Voigt, D., Doellinger, M.: Classification of normal and pathological vocal fold vibrations using phonvibrograms. The Journal of the Acoustical Society of America (2008)

9. Lo, S.C.B., Lin, J.S.J., Freedman, M.T., Mun, S.K.: Application of artificial neural networks to medical image pattern recognition: Detection of clustered microcalcifications on mammograms and lung cancer on chest radiographs. J. VLSI Signal Process. Syst. **18**(3) (1998) 263–274
10. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS'86: Proceedings of the 27th Annual Symposium on Foundations of Computer Science. (1986) 162–167
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing, New York, NY, USA, ACM Press (1987) 218–229
12. Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: A review and open problems. In: Proceedings of the 2001 Workshop on New Security Paradigms, Cloudcroft, NM, ACM Press (September 2001) 13–22
13. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. SIGKDD Explor. Newsl. **4**(2) (2002) 28–34
14. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. SIGKDD Explor. Newsl. **4**(2) (2002) 12–19
15. Vaidya, J., Clifton, C.: Privacy-preserving decision trees over vertically partitioned data. In: The 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, Connecticut, Springer (08 2005)
16. Du, W., Zhan, Z.: Building decision tree classifier on private data. In: CRPIT '14: Proceedings of the IEEE international conference on Privacy, security and data mining, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2002) 1–8
17. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Advances in Cryptology (CRYPTO'00). Volume 1880 of Lecture Notes in Computer Science., Springer-Verlag (2000) 36–53
18. Kantarcioglu, M.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. IEEE Transactions on Knowledge and Data Engineering **16**(9) (2004) 1026–1037
19. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2003) 206–215
20. Vaidya, J., Clifton, C., Zhu, M.: Privacy Preserving Data Mining. Volume 19 of Advances in Information Security. Springer (2006)
21. Wan, L., Ng, W.K., Han, S., Lee, V.C.S.: Privacy-preservation for gradient descent methods. In Berkhin, P., Caruana, R., Wu, X., eds.: KDD, New York, NY, USA, ACM Press (2007) 775–783
22. Han, S., Ng, W.K.: Privacy-preserving self-organizing map. In Song, I.Y., Eder, J., Nguyen, T.M., eds.: DaWaK. Volume 4654 of Lecture Notes in Computer Science., Springer (2007) 428–437
23. Barni, M., Orlandi, C., Piva, A.: A privacy-preserving protocol for neural-network-based computation. In: MM&Sec '06: Proceeding of the 8th workshop on Multimedia and security, New York, NY, USA, ACM Press (2006) 146–151
24. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Interior Representation by Error Propagation. Parallel Distributed Processing **1** (1986) 318–362
25. Yu, H., Vaidya, J.: Secure matrix addition. Technical Report UIOWA CS TR04-04, University of Iowa (November 2004)
26. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press, New York, NY, USA (2004)