Differential Privacy Preservation for Deep Auto-Encoders: An Application of Human Behavior Prediction

NhatHai Phan[1], Yue Wang[2], Xintao Wu[3], Dejing Dou[1]
{haiphan,dou}@cs.uoregon.edu, ywang91@uncc.edu, xintaowu@uark.edu
[1] University of Oregon, USA; [2] University of North Carolina at Charlotte, USA; [3] University of Arkansas, USA

Abstract

In recent years, deep learning has spread beyond both academia and industry with many exciting real-world applications. The development of deep learning has presented obvious privacy issues. However, there has been lack of scientific study about privacy preservation in deep learning. In this paper, we concentrate on the auto-encoder, a fundamental component in deep learning, and propose the *deep private auto-encoder* (dPA). Our main idea is to enforce ϵ-differential privacy by perturbing the objective functions of the traditional deep auto-encoder, rather than its results. We apply the dPA to human behavior prediction in a health social network. Theoretical analysis and thorough experimental evaluations show that the dPA is highly effective and efficient, and it significantly outperforms existing solutions.

Introduction

Deep learning is a timely and promising area of machine learning research. In general, deep learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text (Deng and Yu 2014). With significant success in recent years, the applications of deep learning are being expanded into other fields such as social media (Yuan et al. 2013), social network analysis (Perozzi, Al-Rfou, and Skiena 2014), bioinformatics (Chicco, Sadowski, and Baldi 2014), medicine and healthcare (Song et al. 2014). This presents obvious issues about protecting privacy in deep learning models when they are built on users' personal and highly sensitive data, such as clinical records, user profiles, photo, etc. The present research efforts lack sufficient deep learning techniques that incorporate privacy concerns. Therefore, developing a privacy preserving deep learning model is an urgent demand.

Releasing sensitive results of statistical analysis and data mining while protecting privacy has been studied in the past few decades. One state-of-the-art approach to the problem is ϵ-differential privacy (Dwork et al. 2006), which works by injecting random noise into the released statistical results computed from the underlying sensitive data, such that the distribution of the noisy results is relatively insensitive to any change of a single record in the original dataset. This ensures that the adversary cannot infer any information about any particular record with high confidence (controlled by parameter ϵ), even if the adversary possesses all the remaining tuples of the sensitive data. In this paper, we propose to develop an ϵ-differential privacy preserving deep learning model. The structure of deep learning models, e.g., deep auto-encoders (Bengio 2009), deep belief networks (Hinton, Osindero, and Teh 2006), usually contains multiple layers of neurons. At each layer, they use different objective functions (e.g., cross-entropy error, energy based functions) and algorithms (e.g., contrastive divergence (Hinton 2002), backpropagation) to learn the optimal parameters for output tasks such as classification and prediction. In addition, the designs of deep learning models are varied and dependent on application domains. It is difficult, therefore, to design a unified ϵ-differential privacy solution that covers all deep learning models. *Deep auto-encoders* (dAs) (Bengio 2009) are one of the fundamental deep learning models which have been used in many applications such as healthcare and medicine, natural language processing, etc. (Deng and Yu 2014). In this paper, we aim at deriving an ϵ-differential privacy approach to deep auto-encoders (dAs) for a binomial classification task.

There are several potential solutions for ϵ-differential privacy that target regression, which is similar to the data reconstruction and cross-entropy error functions in deep auto-encoders (Zhang et al. 2012; Chaudhuri and Monteleoni 2008; Chaudhuri, Monteleoni, and Sarwate 2011; Lei 2011; Smith 2011). The most pertinent algorithm that we can adapt in our work is *functional mechanism* (FM) (Zhang et al. 2012). The FM enforces ϵ-differential privacy by perturbing the *objective function* of the optimization problem, e.g., logistic regression, rather than its results. By leveraging the functional mechanism to perturb the objective functions in deep auto-encoders so that the ϵ-differential privacy is preserved, we propose a novel *ϵ-differential **P**rivate **A**uto-encoder* (PA). First, the cross-entropy error functions of the data reconstruction and softmax layer are approximated to polynomial forms by using Taylor Expansion (Arfken 1985). Second, we inject noise into these polynomial forms so that the ϵ-differential privacy is satisfied in the training phases. Third, in the PA, we add a *normalization layer* on top of the hidden layer to protect the ϵ-differential privacy when stacking multiple PAs. Multiple PAs can be stacked on each other to produce a *deep private auto-encoder* (dPA). To evaluate its performance, we apply the dPA model for human behavior prediction in a real health social network. Conducted experimental results demonstrate that the dPA model achieves accurate results with prediction power comparable to the unperturbed results, and it outperforms the existing solutions.

Preliminaries and Related Works

In this section, we briefly revisit the differential privacy definition, Functional Mechanism (Zhang et al. 2012), and formally introduce the deep auto-encoder. Let D be a database that contains n tuples $t_1, t_2, \dots, t_n$ and d+1 attributes $X_1, X_2, \dots, X_d, Y$. For each tuple $t_i = (x_{i1}, x_{i2}, \dots, x_{id}, y_i)$, we assume without loss of generality $\sqrt{\sum_{j=1}^{d} x_{ij}^2} \leq 1$ where $x_{ij} \geq 0$, y_i follows a binomial distribution. Our objective is to construct a deep auto-encoder ρ from D that (i) takes $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})$ as input and (ii) outputs a prediction of y_i that is as accurate as possible. The model function ρ contains a model parameter vector ω. To evaluate whether ω leads to an accurate model, a cost function $f_D(\omega)$ is often used to measure the difference between the original and predicted values of y_i. As the released model parameter ω may disclose sensitive information of D, to protect the privacy, we require that the model training should be performed with an algorithm which satisfies ϵ-*differential privacy*.

Differential Privacy. Differential privacy research has been significantly studied from the theoretical perspective, e.g., (Chaudhuri and Monteleoni 2008; Kifer and Machanavajjhala 2011). There are also extensive studies on the applicability of enforcing differential privacy in some particular analysis tasks, e.g., collaborative recommendation (McSherry and Mironov 2009), logistic regression (Chaudhuri and Monteleoni 2008), publishing contingency tables (Xiao, Wang, and Gehrke 2010), and spectral graph analysis (Wang, Wu, and Wu 2013) in social network analysis. The mechanisms of achieving differential privacy mainly include the classic approach of adding Laplacian noise (Dwork et al. 2006), the exponential mechanism (McSherry and Talwar 2007), and the functional perturbation approach (Chaudhuri and Monteleoni 2008).

Definition 1 *(ϵ-Different Privacy (Dwork et al. 2006)). A randomized algorithm A fulfills ϵ-differential privacy, iff for any two databases D and D' differing at most one tuple, and for any output O of A, we have:*

$$Pr(A(D) = O) \leq e^{\epsilon} Pr(A(D') = O) \qquad (1)$$

The privacy budget ϵ controls the amount by which the distributions induced by two neighboring data sets may differ. Smaller values of ϵ enforce a stronger privacy guarantee of A. A general method for computing an approximation to any function f (on D) while preserving ϵ-differential privacy is the *Laplace mechanism* (Dwork et al. 2006), where the output of f is a vector of real numbers. In particular, the mechanism exploits the global sensitivity of f over any two neighboring data sets (differing at most one record), which is denoted as $S_f(D)$. Given $S_f(D)$, the Laplace mechanism ensures ϵ-differential privacy by injecting noise η into each value in the output of $f(D)$ where η is drawn i.i.d. from Laplace distribution with zero mean and scale $S_f(D)/\epsilon$. In (Shokri and Shmatikov 2015), Shokri et al. proposed a distributed training method to preserve privacy in neural networks. They extended their method to preserve differential privacy for distributed deep neural networks. Their method aims at preserving epsilon-differential privacy for each training epoch of each participant. Even their method is reasonable given the context of distributed deep neural networks; their approach may consume too much privacy budget to ensure model accuracy when the number of training epochs, the number of participants, and the number of shared parameters are large.

Functional Mechanism Revisited. Functional mechanism (Zhang et al. 2012) is an extension of the Laplace mechanism. It achieves ϵ-differential privacy by perturbing the objective function $f_D(\omega)$ and then releasing the model parameter $\overline{\omega}$ that minimizes the perturbed objective function $\overline{f}_D(\omega)$ instead of the original one. The functional mechanism exploits the polynomial representation of $f_D(\omega)$. The model parameter ω is a vector that contains d values $\omega_1, \dots, \omega_d$. Let $\phi(\omega)$ denote a product of $\omega_1, \dots, \omega_d$, namely, $\phi(\omega) = \omega_1^{c_1} \cdot \omega_2^{c_2} \cdots \omega_d^{c_d}$ for some $c_1, \dots, c_d \in \mathbb{N}$. Let $\Phi_j (j \in \mathbb{N})$ denote the set of all products of $\omega_1, \dots, \omega_d$ with degree j, i.e., $\Phi_j = \{\omega_1^{c_1} \cdot \omega_2^{c_2} \cdots \omega_d^{c_d} \big| \sum_{l=1}^{d} c_l = j\}$. By the Stone-Weierstrass Theorem, any continuous and differentiable $f(t_i, \omega)$ can always be written as a polynomial of $\omega_1, \dots, \omega_d$, for some $J \in [0, \infty]$, i.e., $f(t_i, \omega) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \lambda_{\phi t_i} \phi(\omega)$ where $\lambda_{\phi t_i} \in \mathbb{R}$ denotes the coefficient of $\phi(\omega)$ in the polynomial.

For instance, the polynomial expression of the loss function in the linear regression is as follows: $f(\mathbf{x}_i, \omega) = (y_i - \mathbf{x}_i^T \omega)^2 = y_i^2 - \sum_{j=1}^{d} (2 y_i x_{ij}) \omega_j + \sum_{1 \leq j,l \leq d} (x_{ij} x_{il}) \omega_j \omega_l$. We can see that it only involves monomials in $\Phi_0 = \{1\}, \Phi_1 = \{\omega_1, \dots, \omega_d\}$, and $\Phi_2 = \{\omega_i \omega_j \big| i, j \in [1, d]\}$. Each $\phi(\omega)$ has its own coefficient, e.g., for ω_j, its polynomial coefficient $\lambda_{\phi_{t_i}} = -2 y_i x_{ij}$. Similarly, $f_D(\omega)$ can also be expressed as a polynomial of $\omega_1, \dots, \omega_d$.

$$f_D(\omega) = \sum_{j=0}^{J} \sum_{\phi \in \Phi_j} \sum_{t_i \in D} \lambda_{\phi t_i} \phi(\omega) \qquad (2)$$

Lemma 1 *(Zhang et al. 2012) Let D and D' be any two neighboring datasets. Let $f_D(\omega)$ and $f_{D'}(\omega)$ be the objective functions of regression analysis on D and D', respectively. The following inequality holds*

$$\Delta = \sum_{j=1}^{J} \sum_{\phi \in \Phi_j}^{2} \Big\| \sum_{t_i \in D} \lambda_{\phi t_i} - \sum_{t'_i \in D'} \lambda_{\phi t'_i} \Big\|_1$$

$$\leq 2 \max_t \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \|\lambda_{\phi t}\|_1$$

where t_i, t'_i or t is an arbitrary tuple.

To achieve ϵ-differential privacy, $f_D(\omega)$ is perturbed by injecting Laplace noise $Lap(\frac{\Delta}{\epsilon})$ into its polynomial coefficients λ_ϕ, and then the model parameter $\overline{\omega}$ is derived to minimize the perturbed function $\overline{f}_D(\omega)$, where $\Delta = 2\max_t \sum_{j=1}^{J} \sum_{\phi \in \Phi_j} \|\lambda_{\phi t}\|_1$, according to the Lemma 1.

When the polynomial form of an objective function (e.g., logistic regression) contains terms with unbounded degrees, Zhang et al. (Zhang et al. 2012) developed an approximation polynomial form based on Taylor Expansion (Arfken 1985). For instance, given the cost function $f(t_i, \omega)$ of regression analysis, assume that there exist $2m$ functions $f_1, \ldots, f_m$ and $g_1, \ldots, g_m$ such that $f(t_i, \omega) = \sum_{l=1}^{\infty} f_l(g_l(t_i, \omega))$, and each $g_l(t_i, \omega)$ is a polynomial function of ω, where $m \in \mathbb{N}$ is the number of functions f and g. Given the above decomposition of $f(t_i, \omega)$, we can apply the Taylor Expansion (Arfken 1985) on each $f_l(\cdot)$ to obtain the following equation:

$$\widetilde{f}(t_i, \omega) = \sum_{l=1}^{m} \sum_{R=0}^{\infty} \frac{f_l^{(R)}(z_l)}{R!} \big(g_l(t_i, \omega) - z_l\big)^R \tag{3}$$

where each z_l is a real number.

Thus, the objective function $f(D, \omega)$ can be written as a polynomial function, i.e.,

$$\widetilde{f}_D(\omega) = \sum_{i=1}^{|D|} \sum_{l=1}^{m} \sum_{R=0}^{\infty} \frac{f_l^{(R)}(z_l)}{R!} \big(g_l(t_i, \omega) - z_l\big)^R \tag{4}$$

Deep Auto-Encoders. The deep auto-encoder (Bengio 2009) is one of the fundamental models in deep learning (Deng and Yu 2014). An auto-encoder has a layer of input units, and a layer of data reconstruction units fully connected to a layer of hidden units, but no connections within a layer (Fig. 1a). An auto-encoder is trained to encode the input in some representation $\mathbf{x}_i = \{x_{i1}, \ldots, x_{id}\}$ so that the input can be reconstructed from that representation. The auto-encoder prefers to minimize the negative log-likelihood of the reconstruction, given the encoding $\widetilde{\mathbf{x}}_i$,

$$\begin{aligned} RE(t_i, W) &= -\log P(\mathbf{x}_i | \widetilde{\mathbf{x}}_i, W) \\ &= -\sum_{j=1}^{d} \big(x_{ij} \log \widetilde{x}_{ij} + (1 - x_{ij}) \log(1 - \widetilde{x}_{ij})\big) \end{aligned} \tag{5}$$

where W is a weight matrix (i.e., from now on W will be used to denote the parameters instead of ω),

$$\widetilde{\mathbf{x}} = \sigma(hW) \quad , \quad h = \sigma(W^T \mathbf{x}) \tag{6}$$

$\sigma(\cdot)$ is the sigmoid function. The loss function on the whole dataset D is the summation over all the data points $\mathbf{x}$:

$$\begin{aligned} RE(D, W) = \sum_{i=1}^{|D|} RE(\mathbf{x}_i, W) = \sum_{i=1}^{|D|} \sum_{j=1}^{d} \Big[\\ x_{ij} \log(1 + e^{-W_j h_i}) + (1 - x_{ij}) \log(1 + e^{W_j h_i}) \Big] \end{aligned} \tag{7}$$

In the following, we blur the distinction between $\mathbf{x}_i$ and t_i, which means t_i indicates the set of attributes in $\mathbf{x}_i$. t_i and $\mathbf{x}_i$ will be used interchangeably. We can stack multiple auto-encoders to produce a deep auto-encoder. On top of the deep auto-encoder, we add an output layer which includes a single binomial variable to predict Y. The output variable $\hat{y}$ is fully linked to the hidden variables of the highest hidden layer, denoted $h_{(k)}$, by weighted connections $W_{(k)}$, where k is the number of hidden layers in the deep auto-encoder. We use the logistic function as an activation function of $\hat{y}$, i.e., $\hat{y} = \sigma(W_{(k)} h_{(k)})$. A loss function to appropriately deal with the binomial problem is *cross-entropy error*. Let Y_T be a set of labeled data points used to train the model, the cross-entropy error function is given by

$$C(Y_T, \theta) = -\sum_{i=1}^{|Y_T|} \Big(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \Big) \tag{8}$$

Algorithm 1: Pseudo Code of a dPA model

```
1) Derive polynomial approximation of data reconstruction
   function $RE(D,W)$ (Eq. 7), denoted as $\widehat{RE}(D,W)$
2) The function $\widehat{RE}(D,W)$ is perturbed by using functional
   mechanism (FM) (Zhang et al. 2012), the perturbed function
   is denoted as $\overline{RE}(D,W)$
3) Compute $\overline{W} = \arg\min_W \overline{RE}(D,W)$
4) Private Auto-encoder (PA) stacking
5) Derive and perturb the polynomial approximation of
   cross-entropy error $C(\theta)$ (Eq. 8), the perturbed function is
   denoted as $\overline{C}(\theta)$
6) Compute $\overline{\theta} = \arg\min_\theta \overline{C}(\theta)$; Return $\overline{\theta}$
```

We can use the layer-wise unsupervised training algorithm (Bengio et al. 2007) and *back-propagation* to train deep auto-encoders.

Deep Private Auto-Encoder (dPA)

In this section, we formally present our dPA model, a deep auto-encoder model under ϵ-differential privacy. Our algorithm to construct the dPA model includes six main steps (Algorithm 1). We leverage the functional mechanism to enforce ϵ-differential privacy in our dPA model. Therefore, in the first step (Algorithm 1), we derive a polynomial approximation of data reconstruction function $RE(D, W)$ (Eq. 7). The polynomial approximation is denoted as $\widehat{RE}(D, W)$. This is a non-trivial task due to the following key challenges and requirements: 1) The data reconstruction function is complicated (compared with linear or logistic regression analysis); and 2) The error of approximation methods must be bounded and independent of the data size $|D|$. The second requirement guarantees the potential to use unlabeled data in a dPA model. As shown in the next sections, our approximation method theoretically satisfies this requirement.

In the second step, the functional mechanism is used to perturb the approximation function $\widehat{RE}(D, W)$, the perturbed function is denoted as $\overline{RE}(D, W)$. In this step, we introduce a new result of *global sensitivity* computation for auto-encoders. In the third step, we train the model to obtain the optimal perturbed parameters $\overline{W}$ by using gradient descent. That results in private auto-encoders (PAs). In the fourth step, we stack private auto-encoders (PAs) to construct the deep private auto-encoders (dPAs). Before each stacking operation, we introduce a *normalization layer*, denoted as $\mathfrak{h}$ (Fig. 1b), which guarantees all the data assumptions and ϵ-differential privacy will be satisfied when training the next hidden layer. The normalization layer is placed on top of each private auto-encoder. In the fifth step, we introduce new results in deriving and perturbing (i.e., by using functional mechanism) the polynomial approximation of the cross-entropy error $C(\theta)$ (Eq. 8) in the softmax layer for prediction tasks. The perturbed function is denoted as $\overline{C}(\theta)$. At the sixth step, back-propagation algorithm is used to fine-tune all the parameters in the deep private auto-encoders.

In our framework, ϵ-differential privacy is preserved in our dPA model, since it is enforced at every layer and training step. Let us first derive the polynomial approximation form of $RE(D,W)$, denoted $\widehat{RE}(D,W)$, as follows.

Polynomial Approximation. To explain how we apply the Taylor Expansion, and how Equations 3 and 4 are related to $RE(D,W)$, recall the loss function shown in Eq. 7. $\forall j \in \{1,\ldots,d\}$, let f_{1j}, f_{2j}, g_{1j}, and g_{2j} be four functions defined as follows:

$$g_{1j}(t_i, W_j) = W_j h_i; g_{2j}(t_i, W_j) = W_j h_i; f_{1j}(z_j) = \\ x_{ij}\log(1+e^{-z_j}); f_{2j}(z_j) = (1-x_{ij})\log(1+e^{z_j}) \quad (9)$$

Then, we have

$$RE(t_i, W) = \sum_{j=1}^{d}\Big(f_{1j}\big(g_{1j}(t_i,W_j)\big) + f_{2j}\big(g_{2j}(t_i,W_j)\big)\Big)$$

By Equations 3 and 4, we have: $\widetilde{RE}(D,W) =$

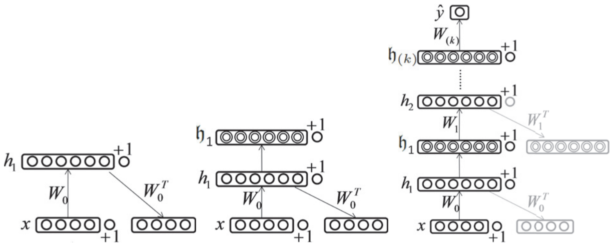
$$= \sum_{i=1}^{|D|}\sum_{j=1}^{d}\sum_{l=1}^{2}\sum_{R=0}^{\infty}\frac{f_{lj}^{(R)}(z_{lj})}{R!}\big(g_{lj}(t_i,W_j) - z_{lj}\big)^R$$

where each z_{lj} is a real number. By setting $z_{lj} = 0$, the above equation can be simplified as:

$$\widetilde{RE}(D,W) = \sum_{i=1}^{|D|}\sum_{j=1}^{d}\sum_{l=1}^{2}\sum_{R=0}^{\infty}\frac{f_{lj}^{(R)}(0)}{R!}\big(W_j h_i\big)^R \quad (10)$$

There are two complications in Eq. 10 that prevent us from applying it for private data reconstruction analysis. First, the equation involves an infinite summation. Second, the term $f_{lj}^{(R)}(0)$ involved in the equation does not have a closed form solution. We address these two issues by presenting an approximation approach that reduces the degree of the summation. Our approximation approach works by truncating the Taylor series in Eq. 10 to remove all polynomial terms with order larger than 2. This leads to a new objective function with low order polynomials as follows:

$$\widehat{RE}(D,W) = \sum_{i=1}^{|D|}\sum_{j=1}^{d}\Big(\sum_{l=1}^{2} f_{lj}^{(0)}(0) \\ + \big(\sum_{l=1}^{2} f_{lj}^{(1)}(0)\big)W_j h_i + \big(\sum_{l=1}^{2}\frac{f_{lj}^{(2)}(0)}{2!}\big)(W_j h_i)^2\Big) \quad (11)$$



(a) Auto-encoder (b) PA model (c) dPA model

Figure 1: Examples of Auto-encoder, Private Auto-encoder, Deep Private Auto-encoder.

Perturbation of Objective Functions. In this section, we employ the functional mechanism to perturb the objective function $\widehat{RE}(D,W)$ (i.e., in Eq. 11) by injecting Laplace noise into its polynomial coefficients. The perturbed function is denoted as $\overline{RE}(D,W)$. Then we derive the model parameter W that minimizes the perturbed function $\overline{RE}(D,W)$. First of all, we need to explore the sensitivity, denoted as Δ, of $\widehat{RE}$ on D. In Eq. 11, $\sum_{l=1}^{2} f_{lj}^{(0)}(0)$, $\sum_{l=1}^{2} f_{lj}^{(1)}(0)$, and $\sum_{l=1}^{2}\frac{f_{lj}^{(2)}(0)}{2!}$ essentially are the polynomial coefficients of the function $\widehat{RE}$ given the database D. To be concise, we denote $\{\lambda_{jt_i}^{(0)}, \lambda_{jt_i}^{(1)}, \lambda_{jt_i}^{(2)}\}$ as a set of the coefficients where $t_i \in D$, $\lambda_{jt_i}^{(0)} = \sum_{l=1}^{2} f_{lj}^{(0)}(0)$, $\lambda_{jt_i}^{(1)} = \sum_{l=1}^{2} f_{lj}^{(1)}(0)$, and $\lambda_{jt_i}^{(2)} = \sum_{l=1}^{2}\frac{f_{lj}^{(2)}(0)}{2!}$. We are now ready to state the following lemma.

Lemma 2 *Let D and D' be any two neighboring databases. Let $\widehat{RE}(D,W)$ and $\widehat{RE}(D',W)$ be the objective functions of auto-encoder learning on D and D' respectively, then the global sensitivity of $\widehat{RE}$ over any two neighboring databases is as follows:*

$$\Delta = 2\max_t \sum_{j=1}^{d}\sum_{R=0}^{2}\|\lambda_{jt}^{(R)}\| \leq d(b + \frac{1}{4}b^2) \quad (12)$$

where b is the number of hidden variables.

We leave the detailed proof of Lemma 2 in the Appendix. We use gradient descent to train the perturbed model $\overline{RE}(D,W)$.

Stacking Private Auto-Encoders. We have presented Private Auto-Encoders (PAs), the ϵ-differential privacy preserving algorithm for auto-encoders. To construct a deep Private Auto-Encoder (dPA) in which ϵ-differential privacy is satisfied, we need to stack multiple PAs on top of each other. The hidden units of the lower layer will be considered as the input of the next PA (Fig. 1). To guarantee that this input to the next PA satisfies our assumption (i.e., $\sqrt{\sum_{j=1}^{b} h_{ij}^2} \leq 1$ may not hold), we add a *normalization layer*, denoted as $\mathfrak{h}$, on top of the PA. Each variable $\mathfrak{h}_{ij}$ can be directly computed from the h_{ij} as follows:

$$\mathfrak{h}_{ij} = \frac{h_{ij} - \gamma_j}{(\varphi_j - \gamma_j)\cdot\sqrt{b}} \quad (13)$$

where γ_j and φ_j denote the minimum and maximum values in the domain of $H_j = \{h_{1j}, \ldots, h_{|D|j}\}$. Now we have $\sqrt{\sum_{j=1}^{b} \mathfrak{h}_{ij}^2} \leq 1$. On top of the dPA model, we add a softmax layer for either classification or prediction. To guarantee that privacy is preserved at every layer, we need to develop an ϵ-differential privacy preserving softmax layer.

Perturbation Softmax Layer. We focus on a binomial prediction problem for the softmax layer. This layer contains a single binomial variable $\hat{y}$ to predict Y (Fig. 1c). A loss function to appropriately deal with this task is the *cross-entropy error*, which is given in Eq. 8. We can preserve the ϵ-differential privacy for the softmax layer by deriving a polynomial approximation of the cross-entropy error function, which will then be perturbed by using the *functional mechanism*. Function $C(Y_T, \theta)$ in Eq. 8 can be rewritten as

$$C(Y_T, \theta) = -\sum_{i=1}^{|Y_T|} \Big(y_i \log(1 + e^{-W_{(k)}\mathfrak{h}_{i(k)}}) + (1 - y_i)\log(1 + e^{W_{(k)}\mathfrak{h}_{i(k)}}) \Big)$$

where $W_{(k)}$ and $\mathfrak{h}_{i(k)}$ are the states of weights W and hidden variables $\mathfrak{h}$ at the k-th normalization layer, the $\mathfrak{h}_{i(k)}$ is derived from the tuple t_i by aggregating the tuple t_i through the structure of the dPA (Fig. 1b). In addition, the four functions in Eq. 9 become:

$$g_1(t_i, W_{(k)}) = W_{(k)}\mathfrak{h}_{i(k)} \quad , \quad g_2(t_i, W_{(k)}) = W_{(k)}\mathfrak{h}_{i(k)}$$
$$f_1(z) = y_i \log(1 + e^{-z}) \quad , \quad f_2(z) = (1 - y_i)\log(1 + e^{z})$$

The polynomial approximation of $C(Y_T, \theta)$ is as follows:

$$\widehat{C}(Y_T, \theta) = \sum_{i=1}^{|Y_T|} \sum_{l=1}^{2} \sum_{R=0}^{2} \frac{f_l^{(R)}(0)}{R!} \big(W_{(k)}\mathfrak{h}_{i(k)}\big)^R \qquad (14)$$

The global sensitivity of the function of $\widehat{C}(Y_T, \theta)$ over Y_T is given in the following lemma.

Lemma 3 *Let Y_T and Y'_T be any two neighboring sets of labeled data points. Let $\widehat{C}(Y_T, \theta)$ and $\widehat{C}(Y'_T, \theta)$ be the objective functions of auto-encoder learning on Y_T and Y'_T respectively, then the global sensitivity of $\widehat{C}$ over Y_T is as:*

$$\Delta_C = |\mathfrak{h}_{(k)}| + \frac{1}{4}|\mathfrak{h}_{(k)}|^2 \qquad (15)$$

where $|\mathfrak{h}_{(k)}|$ is the number of hidden variables in $\mathfrak{h}_{(k)}$.

We leave the proof of Lemma 3 to the Appendix. After computing the sensitivity Δ_C, functional mechanism is used to achieve the differential privacy for the objective function $\widehat{C}(Y_T, \theta)$. The perturbed function is denoted as $\overline{C}(Y_T, \theta)$. The back-propagation algorithm is used to train all the parameters in the deep Private Auto-Encoders (dPAs) that aim at optimizing the function $\overline{C}(Y_T, \theta)$.

Approximation Error Bounds. The following lemma shows the result of how much error our approximation approaches, $\widehat{RE}(D, W)$ (Eq. 11) and $\widehat{C}(Y_T, \theta)$ (Eq. 14), incur. The error only depends on the structure of the function and the number of attributes of the dataset. In addition, the average error of the approximations is always bounded.

Lemma 4 *Given four polynomial functions $\widetilde{RE}(D, W)$, $\widehat{RE}(D, W)$, $\widetilde{C}(Y_T, \theta)$, and $\widehat{C}(Y_T, \theta)$, the average error of the approximations is always bounded as follows:*

$$|\widetilde{RE}(D, \widehat{W}) - \widetilde{RE}(D, \widetilde{W})| \leq \frac{e^2 + 2e - 1}{e(1 + e)^2} \times d \qquad (16)$$

$$|\widetilde{C}(Y_T, \widehat{\theta}) - \widetilde{C}(Y_T, \widetilde{\theta})| \leq \frac{e^2 + 2e - 1}{e(1 + e)^2} \qquad (17)$$

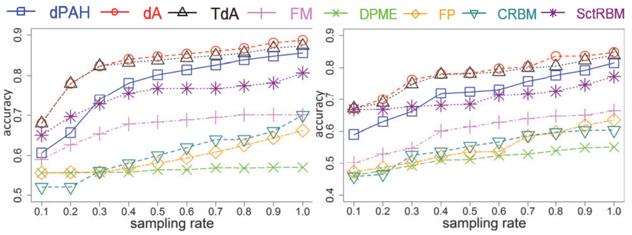
where $\widetilde{W} = \arg\min_W \widetilde{RE}(D, W)$, $\widehat{W} = \arg\min_W \widehat{RE}(D, W)$, $\widetilde{\theta} = \arg\min_\theta \widetilde{C}(Y_T, \theta)$, *and* $\widehat{\theta} = \arg\min_\theta \widehat{C}(Y_T, \theta)$.

We leave the detailed proof of Lemma 4 to the Appendix. Lemma 4 shows that the error of the approximation of the data reconstruction in an auto-encoder is a product of a small constant and the number of attributes d (Eq. 16). It is reasonable, since the more attributes need to be reconstructed, the more errors are injected into the results. The error is completely independent of the number of tuples or data instances. This sufficiently guarantees that our approximation of the auto-encoder can be applied in large datasets. In the experiment section, we will show that our approximation approach leads to accurate results. In addition, the error of the approximation of cross-entropy error function is a small constant. Therefore, the error is independent of the number of labeled data points in this supervised training step.

Experiments

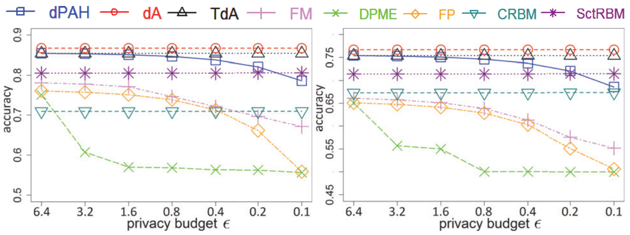
To validate the proposed dPA model, we have developed a dPA-based human behavior prediction model (dPAH) on a real health social network. Our starting observation is that a human behavior is the outcome of behavior determinants such as *self-motivation*, *social influences*, and *environmental events*. This observation is rooted in *human agency in social cognitive theory* (Bandura 1989). In the dPAH model, these human behavior determinants are integrated together.

Our Health Social Network Data were collected from Oct 2010 to Aug 2011 as a collaboration between PeaceHealth Laboratories, SK Telecom Americas, and the University of Oregon to record daily physical activities, social activities (text messages, competitions, etc.), biomarkers, and biometric measures (cholesterol, BMI, etc.) for a group of 254 overweight and obese individuals. Physical activities, including information about the number of walking and running steps, were reported via a mobile device carried by each user. All users enrolled in an online social network, allowing them to friend and communicate with each other. Users' biomarkers and biometric measures were recorded via daily/weekly/monthly medical tests performed at home individually or at our laboratories. In total, we consider three groups of attributes: 1) Behaviors: #joining competitions,



(a) Weekly Dataset (b) Daily Dataset

Figure 2: Prediction accuracy vs. dataset cardinality.



(a) Weekly Dataset (b) Daily Dataset

Figure 3: Prediction accuracy vs. privacy budget ϵ.

#exercising days, #goals set, #goals achieved, $\sum$(distances), avg(speeds); 2) #Inbox Messages: Encouragement, Fitness, Followup, Competition, Games, Personal, Study protocol, Progress report, Technique, Social network, Meetups, Goal, Wellness meter, Feedback, Heckling, Explanation, Invitation, Notice, Technical fitness, Physical; and 3) Biomarkers and Biometric Measures: Wellness Score, BMI, BMI slope, Wellness Score slope.

dPA-based Human Behavior Prediction (dPAH). Given a tuple $t_i, x_{i1}, \ldots, x_{id}$ are the personal attributes and y_i is a binomial parameter which indicates whether a user increases or decreases his/her exercises. To describe the dPAH model, we adjust the notations x_{i1} and y_i a little bit to denote the temporal dimension, and our social network information. Specifically, $x_{ut} = \{x_{1ut}, \ldots, x_{dut}\}$ is used to denote the d attributes of user u at time point t. Meanwhile, y_{ut} is used to denote the status of the user u at time point t. $y_{ut} = 1$ denotes u increases exercises at time t, otherwise $y_{ut} = 0$. The human behavior prediction task is to predict the statuses of all the users in the next time point $t + 1$ given $\mathcal{M}$ past time points $t - \mathcal{M} + 1, \ldots, t$.

Given a user u, to model self-motivation and social influences, we add an aggregation of his/her past attributes and the effects from his/her friends into the activation function of the hidden units at the current time t. The hidden variables h at time t now become

$h_t = \sigma(W_1^T x_{ut} + \hat{a}_{ut})$ where $\hat{a}_{ut} =$

$$= \{\sum_{k=1}^{\mathcal{N}} A_{e,t-k}^T x_{u,t-k} \big| 1 \le e \le b\} + \frac{\eta_u}{|F_u|} \sum_{v \in F_u} \psi_t(v, u)$$

In the above Equation, $\hat{a}_{ut}$ includes a set of dynamic biases for each hidden unit in h_t, i.e., $|h_t| = |\hat{a}_{ut}|$. In addition, $A_{e,t-k}^T$ is a matrix of weight which connects $x_{u,t-k}$ with the e-th hidden variable in h_t. $\psi_t(v, u)$ is the probability that v influences u on physical activity at time t. $\psi_t(v, u)$ is derived from the CPP model (Phan et al. 2014) which is an efficient social influence model in health social networks. F_u is a set of friends of u in the social network. η_u is a parameter which presents the ability to observe the explicit social influences from neighboring users of user u. $\mathcal{N}$ is the number of time points in the past, i.e., $\mathcal{N} < \mathcal{M}$. In essence, the first aggregation term is the effect of user u's past attributes on the current attributes. The second aggregation term captures the effect of social influences from neighboring users on user u. The *environmental events* (i.e., #competitions, #meet-up events) are included in personal attributes. Thus, the effect of environmental events is well embedded into the model.

The parameters and input of our dPAH model are $W = \{W_1, A, \eta_u\}$ and $\mathbf{x} = \{x_{ut}, \cup_k x_{u,t-k}, \frac{\sum_{v \in F_u} \psi_t(v,u)}{|F_u|}\}$. The model includes two PAs and a softmax layer for the human behavior prediction task. The model has been trained on two granular levels of our health social network which are *daily* and *weekly* datasets. Both datasets contain 30 attributes, 254 users, 2,766 messages, 1,383 friend connections, 11,458 competitions, etc. In addition, the daily dataset contains 300 time points, while the weekly dataset contains 48 time points. The number of hidden units and the number of previous time intervals $\mathcal{N}$ are set to 200 and 3, respectively. All the learning rates are set to 10^{-3}.

Competitive Models. We compare the proposed dPAH model with three types of models. **1)** Deep learning models for human behavior prediction. The **CRBM** (Taylor, Hinton, and Roweis 2006) and **SctRBM** (Li et al. 2014; Phan et al. 2015) are competitive models. None of these models enforces ϵ-differential privacy. **2)** *Deep Auto-Encoder* (**dA**) and *Truncated Deep Auto-Encoder* (**TdA**). dA and TdA are two algorithms derived from our analysis in Section 4. They perform analysis but do not enforce ϵ-differential privacy. dA directly outputs the model parameters that minimize the objective functions, and TdA returns the parameters obtained from approximate objective functions with truncated polynomial terms. **3)** Methods for regression analysis under ϵ-differential privacy. This experimental evaluation includes 3 regression analysis approaches which are under ϵ-differential privacy, namely, *Functional Mechanism* (**FM**) (Zhang et al. 2012), **DPME** (Lei 2011), and *Filter-Priority* (**FP**) (Cormode 2011). FM and DPME are the state-of-the-art methods for regression analysis under ϵ-differential privacy, while FP is an ϵ-differentially private technique for generating synthetic data that can also be used for regression tasks. For FM, DPME and FP, we use the implementations provided by their respective authors, and we set all internal parameters to their recommended values.

Accuracy vs. Dataset Cardinality. Fig. 2 shows the prediction accuracy of each algorithm as a function of the dataset cardinality. In this experiment, we vary the size of $\mathcal{M}$, which also can be considered as the sampling rate of the dataset. ϵ is 1.0 in this experiment. In both datasets, there is a gap between the prediction accuracy of dPAH and that of dA and Truncated one TdA, but the gap gets smaller rapidly with the increase of dataset cardinality. In addition, the performance of FM, FP, and DPME improves with the dataset cardinality, which is consistent with the theoretical results in (Zhang et al. 2012; Lei 2011; Cormode 2011). Nevertheless, even when we use all tuples in the dataset, the accuracies of FM, FP, and DPME are still lower than that of dPAH, dA, and TdA. With very small sampling rate, the performance of the dPAH model is slightly lower than the SctRBM, which is a non-privacy-enforcing deep learning model for human behavior prediction. However, the dPAH significantly is better than the SctRBM when the sampling rate goes just a bit higher, i.e., > 0.3. This is a significant result, since 0.3 is a small sampling rate.

Accuracy vs. Privacy Budget. Fig. 3 plots the prediction accuracy of each model as a function of the privacy budget ϵ. The prediction accuracies of dA, TdA, CRBM, and SctRBM remain unchanged for all ϵ. This is because none of them enforces ϵ-differential privacy. Since a smaller ϵ requires a larger amount of noise to be injected, the other four models incur higher inaccurate prediction results when ϵ decreases. dPAH outperforms FM, FP, and DPME in all cases. In addition, it is relatively robust against the change of ϵ. In fact, the dPAH model is competitive even with privacy non-enforcing models such as CRBM and SctRBM.

Conclusions and Future Works

We have developed an approach for differentially private deep auto-encoders. Our approach conducts both sensitivity analysis and noise insertion on the data reconstruction and the cross-entropy error objective functions. The proposed dPA model achieves more accurate prediction results, when the objective functions can be represented as finite polynomials. Our experiments in an application of human behavior prediction in real health social networks validate our theoretical results, and demonstrate the effectiveness of our approach. Our work can be extended on the following directions. First, our current mechanism has the potential to be extended to different deep learning models; since it can be applied to each layer, and stacked. Second, we plan to study whether there may exist alternative analytical tools which can be used to better approximate the objective functions. Third, it is worthy to study how we might be able to extract private information from deep neural networks.

Acknowledgment. This work is supported by the NIH grant R01GM103309 to the SMASH project. Wu is also supported by NSF grant 1502273 and 1523115. Dou is also supported by NSF grant 1118050. We thank Xiao Xiao and Rebeca Sacks for their contributions.

References

Apostol, T. 1967. *Calculus*. John Wiley & Sons.

Arfken, G. 1985. In *Mathematical Methods for Physicists (Third Edition)*. Academic Press.

Bandura, A. 1989. Human agency in social cognitive theory. *The American Psychologist*.

Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H.; Montreal, U. D.; and Quebec, M. 2007. Greedy layer-wise training of deep networks. In *NIPS*.

Bengio, Y. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.* 2(1):1–127.

Chaudhuri, K., and Monteleoni, C. 2008. Privacy-preserving logistic regression. In *NIPS'08*, 289–296.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *J. Mach. Learn. Res.* 12.

Chicco, D.; Sadowski, P.; and Baldi, P. 2014. Deep autoencoder neural networks for gene ontology annotation predictions. In *ACM BCB'14*, 533–540.

Cormode, G. 2011. Personal privacy vs population privacy: learning to attack anonymization. In *KDD'11*, 1253–1261.

Deng, L., and Yu, D. 2014. Deep learning: Methods and applications. Technical Report MSR-TR-2014-21.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. *Theory of Cryptography* 265–284.

Hinton, G.; Osindero, S.; and Teh, Y.-W. 2006. A fast learning algorithm for deep belief nets. *Neural Comput.* 18(7):1527–1554.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14(8).

Kifer, D., and Machanavajjhala, A. 2011. No free lunch in data privacy. In *SIGMOD'11*, 193–204.

Lei, J. 2011. Differentially private m-estimators. In *NIPS*, 361–369.

Li, X.; Du, N.; Li, H.; Li, K.; Gao, J.; and Zhang, A. 2014. A deep learning approach to link prediction in dynamic networks. In *SIAM'14*, 289–297.

McSherry, F., and Mironov, I. 2009. Differentially Private Recommender Systems. In *KDD'09*. ACM.

McSherry, F., and Talwar, K. 2007. Mechanism design via differential privacy. In *FOCS '07*, 94–103.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD'14*, 701–710.

Phan, N.; Dou, D.; Xiao, X.; Piniewski, B.; and Kil, D. 2014. Analysis of physical activity propagation in a health social network. In *CIKM'14*, 1329–1338.

Phan, N.; Dou, D.; Piniewski, B.; and Kil, D. 2015. Social restricted boltzmann machine: Human behavior prediction in health social networks. In *ASONAM'15*, 424–431.

Shokri, R., and Shmatikov, V. 2015. Privacy-preserving deep learning. In *CCS'15*, 1310–1321.

Smith, A. 2011. Privacy-preserving statistical estimation with optimal convergence rates. In *STOC'11*, 813–822.

Song, Y.; Ni, D.; Zeng, Z.; He, L.; et al. 2014. Automatic vaginal bacteria segmentation and classification based on superpixel and deep learning. *Journal of Medical Imaging and Health Informatics* 4(5):781–786.

Taylor, G.; Hinton, G.; and Roweis, S. 2006. Modeling human motion using binary latent variables. In *NIPS'06*, 1345–1352.

Wang, Y.; Wu, X.; and Wu, L. 2013. Differential privacy preserving spectral graph analysis. In *PAKDD (2)*, 329–340.

Xiao, X.; Wang, G.; and Gehrke, J. 2010. Differential privacy via wavelet transforms. In *ICDE'10*, 225–236.

Yuan, Z.; Sang, J.; Liu, Y.; and Xu, C. 2013. Latent feature learning in social media network. In *ACM MM'13*, 253–262.

Zhang, J.; Zhang, Z.; Xiao, X.; Yang, Y.; and Winslett, M. 2012. Functional mechanism: regression analysis under differential privacy. *PVLDB* 5(11):1364–1375.

Appendices

Proof of Lemma 2

Proof 1 *Assume that D and D' differ in the last tuple. Let t_n (t'_n) be the last tuple in D (D'). Then, $\Delta = \sum_{j=1}^{d} \sum_{R=0}^{2} \left\| \sum_{t_i \in D} \lambda_{jt_i}^{(R)} - \sum_{t'_i \in D'} \lambda_{jt'_i}^{(R)} \right\| = \sum_{j=1}^{d} \sum_{R=0}^{2} \|\lambda_{jt_n}^{(R)} - \lambda_{jt'_n}^{(R)}\|$. We can show that $\lambda_{jt_n}^{(0)} = \sum_{l=1}^{2} f_{lj}^{(0)}(0) = x_{nj} \log 2 + (1 - x_{nj}) \log 2 = \log 2$. Similarly, we can show that $\lambda_{jt'_n}^{(0)} = \log 2$. As a result, $\lambda_{jt_n}^{(0)} = \lambda_{jt'_n}^{(0)}$. Therefore*

$$\begin{aligned}
\Delta &= \sum_{j=1}^{d} \sum_{R=0}^{2} \|\lambda_{jt_n}^{(R)} - \lambda_{jt'_n}^{(R)}\| = \sum_{j=1}^{d} \sum_{R=1}^{2} \|\lambda_{jt_n}^{(R)} - \lambda_{jt'_n}^{(R)}\| \\
&\leq \sum_{j=1}^{d} \sum_{R=1}^{2} \left(\|\lambda_{jt_n}^{(R)}\| + \|\lambda_{jt'_n}^{(R)}\|\right) \leq 2 \max_t \sum_{j=1}^{d} \sum_{R=1}^{2} \|\lambda_{jt}^{(R)}\| \\
&\leq 2 \max_t \Big[\sum_{j=1}^{d} (\frac{1}{2} - x_{tj}) \sum_{e=1}^{m} h_{te} + \sum_{j=1}^{d} \big(\frac{1}{8} \sum_{e,q} h_{te} h_{tq} \big) \Big] \\
&\leq 2 (\frac{1}{2} d \times b + \frac{1}{8} d \times b^2) = d(b + \frac{1}{4} b^2)
\end{aligned}$$

where h_{te} is the state of e-th hidden variable derived from the tuple t, i.e., $h = \sigma(W^T \mathbf{x})$.

Proof of Lemma 3

Proof 2 *By applying Lemma 2, we can compute the sensitivity Δ_C of $\widehat{C}(Y_T, \theta)$ on the set of labeled data points Y_T as follows: $\Delta_C \leq 2 \max_t \big[\sum_{j=1}^{|\mathfrak{h}_{(k)}|} (\frac{1}{2} - y_i) + \frac{1}{8} \sum_{e,q} \mathfrak{h}_{te(k)} \mathfrak{h}_{tq(k)} \big] \leq |\mathfrak{h}_{(k)}| + \frac{1}{4} |\mathfrak{h}_{(k)}|^2$, where $\mathfrak{h}_{te(k)}$ is the state of the e-th hidden variable at the k-th normalization layer, and it is derived from the tuple t. $|\mathfrak{h}_{(k)}|$ is the number of hidden variables in $\mathfrak{h}_{(k)}$.*

Proof of Lemma 4

Proof 3 *Let $\widetilde{W} = \arg\min_W \widetilde{RE}(D, W)$ and $\widehat{W} = \arg\min_W \widehat{RE}(D, W)$, $U = \max_W \big(\widetilde{RE}(D, W) - \widehat{RE}(D, W) \big)$ and $S = \min_W \big(\widetilde{RE}(D, W) - \widehat{RE}(D, W) \big)$. We have that $U \geq \widetilde{RE}(D, \widehat{W}) - \widehat{RE}(D, \widehat{W})$ and $\forall W^* : S \leq \widetilde{RE}(D, W^*) - \widehat{RE}(D, W^*)$. Therefore, we have*

$\widetilde{RE}(D, \widehat{W}) - \widehat{RE}(D, \widehat{W}) - \widetilde{RE}(D, W^*) + \widehat{RE}(D, W^*) \leq U - S \Leftrightarrow \widetilde{RE}(D, \widehat{W}) - \widetilde{RE}(D, W^*) \leq U - S + \big(\widehat{RE}(D, \widehat{W}) - \widehat{RE}(D, W^*) \big)$.

In addition, $\widehat{RE}(D, \widehat{W}) - \widehat{RE}(D, W^*) \leq 0$, so $\widetilde{RE}(D, \widehat{W}) - \widetilde{RE}(D, W^*) \leq U - S$. If $U \geq 0$ and $S \leq 0$ then we have:

$$|\widetilde{RE}(D, \widehat{W}) - \widetilde{RE}(D, W^*)| \leq U - S \tag{18}$$

Eq. 18 holds for every W^*. Therefore, it still holds for $\widetilde{W}$. Eq. 18 shows that the error incurred by truncating the Taylor series approximate function depends on the maximum and minimum values of $\widetilde{RE}(D, W) - \widehat{RE}(D, W)$. To quantify the magnitude of the error, we first rewrite $\widetilde{RE}(D, W) - \widehat{RE}(D, W)$ as: $\widetilde{RE}(D, W) - \widehat{RE}(D, W) = \sum_{j=1}^{d} \big(\widetilde{RE}(D, W_j) - \widehat{RE}(D, W_j) \big) = \sum_{j=1}^{d} \Big(\sum_{i=1}^{|D|} \sum_{l=1}^{2} \sum_{R=3}^{\infty} \frac{f_{lj}^{(R)}(z_{lj})}{R!} \big(g_{lj}(t_i, W_j) - z_{lj} \big)^R \Big)$.

To derive the minimum and maximum values of the function above, we look into the remainder of Taylor Expansion for each j. Let $z_j \in [z_{lj} - 1, z_{lj} + 1]$. According to the well-known result (Apostol 1967), $\frac{1}{|D|} \big(\widetilde{RE}(D, W_j) - \widehat{RE}(D, W_j) \big)$ must be in the interval $\Big[\sum_l \frac{\min_{z_j} f_{lj}^{(3)}(z_j)(z_j - z_{lj})^3}{6}, \sum_l \frac{\max_{z_j} f_{lj}^{(3)}(z_j)(z_j - z_{lj})^3}{6} \Big]$. If $\sum_l \frac{\max_{z_j} f_{lj}^{(3)}(z_j)(z_j - z_{lj})^3}{6} \geq 0$ and $\sum_l \frac{\min_{z_j} f_{lj}^{(3)}(z_j)(z_j - z_{lj})^3}{6} \leq 0$, then we have that $|\frac{1}{|D|} \big(\widetilde{RE}(D, W) - \widehat{RE}(D, W) \big)| \leq \sum_{j=1}^{d} \sum_l \frac{\max_{z_j} f_{lj}^{(3)}(z_j)(z_j - z_{lj})^3 - \min_{z_j} f_{lj}^{(3)}(z_j)(z_j - z_{lj})^3}{6}$. This analysis applies to the case of an auto-encoder as follows. First, for the functions $f_{1j}(z_j) = x_{ij} \log(1 + e^{-z_j})$ and $f_{2j}(z_j) = (1 - x_{ij}) \log(1 + e^{z_j})$, we have $f_{1j}^{(3)}(z_j) = \frac{2 x_{ij} e^{z_j}}{(1+e^{z_j})^3}$ and $f_{2j}^{(3)}(z_j) = (1 - x_{ij}) \frac{e^{-z_j}(e^{-z_j} - 1)}{(1+e^{-z_j})^3}$. It can be verified that $\arg\min_{z_j} f_{1j}^{(3)}(z_j) = \frac{-2e}{(1+e)^3} < 0$, $\arg\max_{z_j} f_{1j}^{(3)}(z_j) = \frac{2e}{(1+e)^3} > 0$, $\arg\min_{z_j} f_{2j}^{(3)}(z_j) = \frac{1-e}{e(1+e)^3} < 0$, and $\arg\max_{z_j} f_{2j}^{(3)}(z_j) = \frac{e(e-1)}{(1+e)^3} > 0$. Thus, the average error of the approximation is at most

$|\widetilde{RE}(D, \widehat{W}) - \widetilde{RE}(D, \widetilde{W})| \leq \Big[\big(\frac{2e}{(1+e)^3} - \frac{-2e}{(1+e)^3} \big) + \big(\frac{e(e-1)}{(1+e)^3} - \frac{1-e}{e(1+e)^3} \big) \Big] \times d = \frac{e^2 + 2e - 1}{e(1+e)^2} \times d$.

Therefore, Eq. 16 holds. Eq. 17 can be proved in the similar way. The main difference is that there is only one output variable which is $\hat{y}$ in the function $\widehat{C}(Y_T, \theta)$. Meanwhile, there are d output variables (i.e., $|\widetilde{\mathbf{x}}| = d$) in the data reconstruction function $\widehat{RE}(D, W)$. Thus, by replacing the functions $\widetilde{RE}(D, W)$ and $\widehat{RE}(D, W)$ with $\widetilde{C}(Y_T, \theta)$ and $\widehat{C}(Y_T, \theta)$ in the proof of Eq. 16, we obtain that

$|\widetilde{C}(Y_T, \widehat{\theta}) - \widetilde{C}(Y_T, \widetilde{\theta})| \leq \Big[\big(\frac{2e}{(1+e)^3} - \frac{-2e}{(1+e)^3} \big) + \big(\frac{e(e-1)}{(1+e)^3} - \frac{1-e}{e(1+e)^3} \big) \Big] = \frac{e^2 + 2e - 1}{e(1+e)^2}$.