Privacy Preserving Techniques for Speech Processing

Manas A. Pathak

December 1, 2010

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

PhD Thesis Proposal

Abstract

Speech is perhaps the most private form of personal communication but current speech processing techniques are not designed to preserve the privacy of the speaker and require complete access to the speech recording. We propose to develop techniques for speech processing which do preserve privacy. While our proposed methods can be applied to a variety of speech processing problems and also generally to problems in other domains, we focus on the problems of keyword spotting, speaker identification, speaker verification, and speech recognition.

Each of these applications involve the two separate but overlapping problems of private classifier evaluation and private classifier release. Towards the former, we study designing privacy preserving protocols using primitives such as homomorphic encryption, oblivious transfer, and blind and permute in the framework of secure multiparty computation (SMC). Towards the latter, we study the differential privacy model, and techniques such as exponential method and sensitivity method for releasing differentially private classifiers trained on private data.

We summarize our preliminary work on the subject including techniques such as SMC protocols for eigenvector computation, isolated keyword recognition and differentially private release mechanisms for large margin Gaussian mixture models, and classifiers trained from data belonging to multiple parties. Finally, we discuss the proposed directions of research including techniques such as training differentially private hidden Markov models, and a multiparty framework for differentially private classification by data perturbation, and protocols for applications such as privacy preserving music matching and keyword spotting, speaker verification and identification.

Contents

Chapter 1

Introduction

Speech is perhaps the most private form of personal communication. A sample of a person's voice contains information not only about the message but also about the person's gender, accent, nationality, the emotional state. Therefore, no one would like to have their voice being recorded without consent through eavesdropping or wiretaps – in fact, such activities are considered illegal in most situations. Yet, current speech processing techniques such as speaker identification, speech recognition are not designed to preserve the privacy of the speaker and require complete access to the speech recording.

We propose to develop privacy-preserving techniques for speech processing. There have been privacy preserving techniques proposed for a variety of learning and classification tasks, such as multiple parties performing decision trees [Vaidya et al., 2008a], clustering [Lin et al., 2005], association rule mining [Kantarcioglu and Clifton, 2004], naive Bayes classification [Vaidya et al., 2008b], support vector machines [Vaidya et al., 2008c] and rudimentary computer vision applications [Avidan and Butman, 2006], but there has been little work on techniques for privacy preserving speech processing. While our proposed techniques can be applied to a variety of speech processing problems and also generally to problems in other domains, we focus on the following problems where we believe privacy-preserving solutions are of direct interest.

1. **Keyword spotting.** Keyword spotting systems detect if a specified keyword has occurred in a recording. It is therefore useful if the the system can detect only the presence of the keyword without having access to the speech recording. Such a system would find applications in surveillance applications and pass-phrase detection. It would also enable privacy preserving speech mining applications for extracting statistics about the occurrence of specific keywords in an collection of recordings, while not learning anything else about the speech data.

2. **Speaker Identification.** Speaker identification systems attempt to identify which, if any, of a specified set of speakers has spoken into the system. In many situations, it would be useful to be permit speaker identification without providing access to any other information in the voice. For instance, a security agency may be able to detect if a particular speaker has spoken in a phone conversation without being able to discover who else spoke or what was spoken in the audio.

3. **Speaker Verification.** Speaker verification systems determine if a speaker is indeed who the speaker claims to be. Users may want the system not to have access to their voice and therefore the identity. In text-dependent verification systems, users may not want the system to be able to discover their pass-phrase.

4. **Speech Recognition.** Speech recognition is the problem of automatically transcribing the text from the speech recording. While the use of cloud based online services has been prevalent for many tasks, the private nature of speech data is a major stumbling block towards creating such a speech recognition service. To overcome this, it is desirable to have a privacy preserving speech recognition system which can perform recognition without having access to the speech data.

It should be noted that we are not developing new algorithms which achieve and extend the state of the art performance of the above applications. We are interested in creating privacy preserving frameworks for existing algorithms.

The proposed technology has broad implications not just in the area of speech processing, but to society at large. As voice technologies proliferate, people have increasing reason to distrust them. With increasing use of speech recognition based user interfaces, speaker verification based authentication systems, and automated systems for ubiquitous applications, from routing calls to purchasing airline tickets, the ability of malicious entities to capture and misuse a person's voice has never been greater, and this threat is only expected to increase. The fallout of such misuse can pose a far greater threat than mere loss of privacy but have severe economic and social impacts as well. This proposal represents a proactive effort at developing technologies to secure voice processing systems to prevent such misuse.

1.1 Outline

We first review the basics of speech processing in Chapter 2 and privacy in Chapter 3. Each of the problems mentioned above broadly involves two separate but overlapping aspects: function computation – training and evaluation of classifiers over private speech data and function release – publishing classifiers trained over private speech data. Towards the former problem, we investigate cryptographic protocols in the secure multiparty computation framework (Section 3.1) and for the latter problem we investigate differentially private release mechanisms (Section 3.2). We discuss our preliminary work towards this direction in Chapter 4 and the proposed work in Chapter 5.

Chapter 2

Speech Applications and Privacy

We first review some of the basic building blocks used in speech processing systems. Almost all speech processing techniques follow a two-step process of signal parameterization followed by classification. This is shown in Figure 2.1.

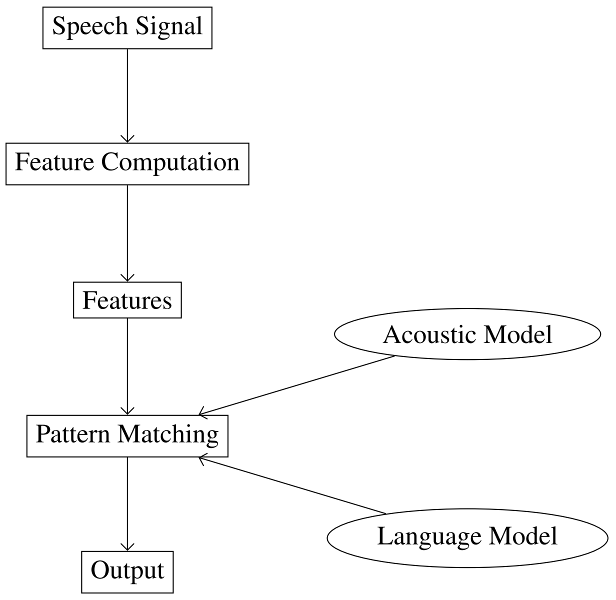

Figure 2.1: Work flow of a speech processing system.

2.1 Basic Tools

2.1.1 Signal Parameterization

The most common parametrization for speech is the mel-frequency cepstral coefficients (MFCC) [Davis and Mermelstein, 1980]. In this representation, we sample the speech signal at a high frequency and take the Fourier transform of each short time window. This is followed by de-correlating the spectrum using a cosine transform, then taking the most significant coefficients.

If X is a vector of signal samples, F is the Fourier transform in matrix form, M is the set of Mel filters represented as a matrix, and D is a DCT matrix, MFCC feature vectors can be computed as $D\log(M((FX)\cdot\text{conjugate}(FX)))$.

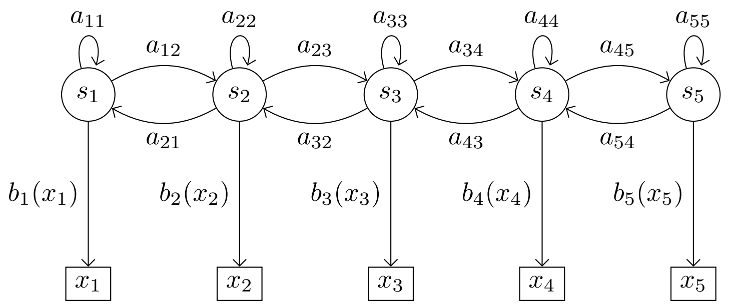

Figure 2.2: An example of a 5-state HMM.

2.1.2 Gaussian Mixture Models

Gaussian mixture model (GMM) is a commonly used generative model for density estimation in speech and language processing. The probability of each class generating an example is modeled as a mixture of Gaussian distributions. For problems such as speaker identification, we have one utterance x which we wish to classify by the class label $y_i \in 1, \ldots, K$ representing a set of speakers. If the mean vector and covariance matrix of the j^{th} Gaussian in the class y_i are μ_{ij} and Σ_{ij}, respectively, for an observation x, we have $P(x|y_i) = \sum_j w_{ij}\mathcal{N}(\mu_{ij}, \Sigma_{ij})$, where w_{ij} are the mixture coefficients. The above mentioned parameters can be computed using the expectation-maximization (EM) algorithm.

2.1.3 Hidden Markov Models

A hidden Markov model (HMM) (Fig. 2.2), can be thought of as an example of a Markov model in which the state is not directly visible but output of each state can be observed. The outputs are also referred to as observations. Since observations depend on the hidden state, an observation reveals information about the underlying state.

A hidden Markov model is defined as a triple $M = (A, B, \Pi)$, in which

- $A = (a_{ij})$ is the state transition matrix. Thus, $a_{ij} = \Pr\{q_{t+1} = S_j | q_t = S_i\}, 1 \leq i, j \leq N$, where $\{S_1, S_2, ..., S_N\}$ is the set of states and q_t is the state at time t.
- $B = (b_j(v_k))$ is the matrix containing the probabilities of the observations. Thus, $b_j(v_k) = \Pr\{\mathbf{x}_t = v_k | q_t = S_j\}, 1 \leq j \leq N, 1 \leq k \leq M$, where $v_k \in V$ which is the set of observation symbols, and $\mathbf{x}_t$ is the observation at time t.
- $\Pi = (\pi_1, \pi_2, ..., \pi_N)$ is the initial state probability vector, that is, $\pi_i = \Pr\{q_1 = S_i\}, i = 1, 2, ..., N$.

Depending on the set of observation symbols, we can classify HMMs into those with discrete outputs and those with continuous outputs. In speech processing applications, we consider HMMs with continuous outputs where each the observation probabilities of each state is modeled using a GMM. Such a model is typically used to model the sequential audio data frames representing the utterance of one word.

For a given sequence of observations $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$ and an HMM $\lambda = (A, B, \Pi)$, one problem of interest is to efficiently compute the probability $P(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T | \lambda)$. A well known solution to this problem is the forward-backward algorithm.

2.2 Speech Processing Applications

We first introduce some terminology that we will employ in the rest of this document. The term speech processing refers to pattern classification or learning tasks performed on voice data. A privacy preserving transaction is one where no party learns anything about the others data. In the context of a speech processing system, this means that the system does not learn anything about the user's speech and the user does not learn anything of the internal parameters used by the system. We often also refer to privacy-preserving operations as secure. We expand the conventional definition of system to include the entity in charge of it who could log data and partial results for analysis. Current voice processing technologies are not designed to preserve the privacy of the speaker. Systems need complete access to the speech recording which is usually in the parameterized form. The only privacy ensured is the loss of information effected by the parametrization. Standard parametrizations can be inverted to obtain an intelligible speech signal and provide little protection of privacy. Yet, there are many situations in which voice processing needs to be performed while preserving the privacy of subjects' voices and processing may need to be performed without having access to the voice. Here, by "access to voice" we refer to having access to any form of the speech that can be converted to an intelligible signal, or from which information about the speaker or what was spoken could be determined.

There are three basic application areas where privacy preserving speech processing finds application.

1. **Biometric Authentication.**

   As a person's voice is characteristic to the individual, speech is widely used in biometric authentication systems. This is also an important privacy concern for the user as the system always has access to the speech recording. It is also possible for an adversary to break into the system to gain unauthorized access using pre-recorded voice. This can be prevented by using privacy preserving speaker verification.

2. **Mining Speech Data.**

   Corporations and call centers often have large quantities of voice data from which they may desire to detect patterns. However, privacy concerns prevent them from providing access to external companies that can mine the data for patterns. Privacy preserving speaker identification and keyword spotting techniques can enable the outside company to detect patterns without learning anything else about the data.

3. **Recognition.**

   Increasingly, speech recognition systems are deployed in a client server framework, where the client has the audio recording and the server has a trained model. In this case, privacy is important as the server has complete access to the audio which is being used for recognition and is often the bottleneck in using such speech recognition systems in contexts where the speech contains sensitive information. Similarly, it is also useful to develop techniques for training the model parameters.

The objective of our proposal is to develop privacy preserving techniques for some of the key voice processing technologies mentioned above. Specifically, we propose to develop privacy preserving techniques for speaker identification, speaker verification, and keyword spotting. Note that our goal is not the development of more accurate and scalable speech processing algorithms; we focus will be to restructure the computations of current algorithms and embedding them within privacy frameworks such that the operations are preserve privacy. Where necessary, we will develop alternative algorithms or implementations that are inherently more amenable to having their computations restructured in this manner. We review the above mentioned speech processing applications along with the relevant privacy issues.

2.2.1 Speaker Verification and Identification

In speaker verification, we try to ascertain if a user is who he or she claims to be. Speaker verification systems can be text dependent, where the speaker utters a specific pass phrase and the system verifies it by comparing the utterance with the version recorded initially by the user. Alternatively, speaker verification can be text independent, where the speaker is allowed to say anything and the system only determines if the given voice sample is close to the speaker's voice. Speaker identification is a related problem in which we identify if a speech sample is spoken by any one of the speakers from our pre-defined set of speakers. The techniques employed in the two problems are very similar, enrollment data from each of the speakers are used to build statistical or discriminative models for the speaker which are employed to recognize the class of a new audio recording.

Basic speaker verification and identification systems use a GMM classifier model trained over the voice of the speaker. We typically use MFCC features of the audio data instead of the original samples as they are known to provide better accuracy for speech classification. In case of speaker verification, we train a binary GMM classifier using the audio samples of the speaker as one class and a universal background noise (UBM) model as another class [Campbell, 2002]. UBM is trained over the combined speech data of all other users. Due to the sensitive nature of their use in authentication systems, speaker verification classifiers need to be robust to false positives. In case of doubt about the authenticity of a user, the system should choose to reject. In case of speaker identification, we also use the UBM to categorize a speech sample as not being spoken by anyone from the set of speakers.

In practice, we need a lot of data from one speaker to train an accurate speaker classification model and such data is difficult to acquire. Towards this, Reynolds et al. [2000] proposed techniques for maximum *a posteriori* adaptation to derive speaker models from the UBM. These adaptation techniques have been extended by constructing "supervectors" consisting of the stacked means of the mixture components [Kenny and Dumouchel, 2004]. The supervector formulation has also been used with support vector machine (SVM) classification methods. Campbell et al. [2006] derive a linear kernel based upon an approximation to the KL divergence between the two GMM models. It might be noted that apart from the conventional generative GMM models, large margin GMMs [Sha and Saul, 2006] can also be used for speaker verification. A variety of other classification algorithms based on HMMs and SVMs have been proposed for speaker verification and speaker identification [Reynolds, 2002; Hatch and Stolcke, 2006].

Privacy Issues

There are different types of privacy requirements while performing speaker verification or identification in terms of training the classifier and evaluating a classifier.

1. **Training.** The basic privacy requirement here is that system should be able to train the speaker classification model without being able to observe the speech data belonging to the users. This problem is an example of secure multiparty computation (Section 3.1), and protocols can be designed for this using primitives such as homomorphic encryption. The UBM is trained over speech data acquired from different users. The individual contribution of every user is private and the system should not have access to the speech recording. Additionally, even if the model is trained privately, the system should not be able to make any inference about the training data by analyzing the model. The differential privacy framework (Section 3.2) provides a probabilistic guarantee in such a setting. This will allow the participants to share their data freely without any privacy implications.

   In case of speaker verification, we need to train one classification model per speaker. In this case, the privacy requirement is that the system should not know which model is currently being trained, *i.e.*, it should not be able to know the identity of any of speakers in the system.

2. **Testing.** Once the system has trained the model, we need to perform classification on new audio data. Here, the privacy requirement is that the system should not be able to observe the test data. Also, as the classification model belonging to the system might be trained over valuable training data, it is not desirable for the system to release the model parameters to the party with the test audio data. Finally, in both speaker verification and identification, the system should be oblivious about the output of the classification. This problem is also an example of secure multiparty computation and can be addressed using the appropriate primitives.

2.2.2 Music Matching

Music matching falls under the broad category of music retrieval or audio mining applications. In a client-server model, we require to compare the input audio snippet of the client: Alice to a collection of longer audio recordings belonging to the server: Bob. Music matching involves finding an audio recording of a song from a central database which most closely matches a given audio snippet. When a match is found, information about the song such as the title, artist, and album are transfered back to the user.

A simple algorithm to perform this matching is by computing the correlation between the music signals. This method is fairly robust to noise in the snippets. Alternatively, if we are dealing with snippets which are also a copy of the original recording, we can directly compare a binary fingerprint of the snippet more efficiently. This technique is used in commercial systems such as [Wang, 2003]. To compare a short snippet to an audio recording of a song, we need to perform a sliding window comparison for all samples of the recording, as shown in Figure 2.3. Bob needs to compare every frame of every song in the collection with the snippet and choose the song with the highest score.

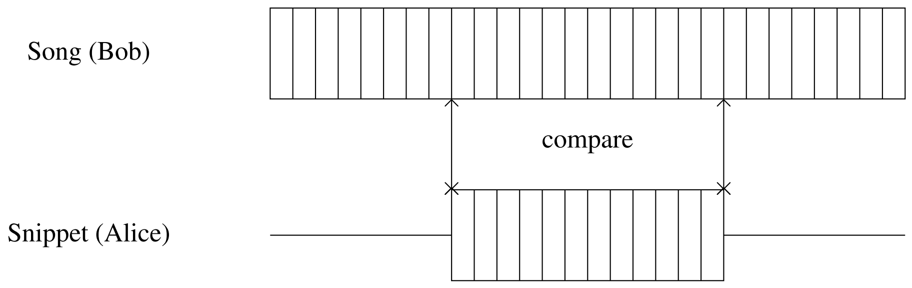

Figure 2.3: Sliding window comparison of snippet frames and song frames.

Privacy Issues

Subsequently, if we need to perform music matching with privacy, our main requirement is that we need to do it without the server observing the snippet provided by the client. Such a system would be very valuable in commercial as well as military settings as discussed above. A trivial and completely impractical solution to this problem is that Bob sends his entire music collection to Alice. Instead, we are interested in solutions in which Alice and Bob participate in a protocol which uses secure multiparty computation primitives (see Section 3.1) to perform the matching securely. Alice will give to Bob an encryption of her audio snippet and Bob will be able to perform the matching operations privately and obtain all the sliding window scores in an encrypted form. Alice and Bob can then participate in an secure maximum finding protocol to obtain the song with the highest score without the knowledge of Bob. Even doing this tends to be computationally expensive, as it is performing operations on encrypted data is costly and still requires some amount of data transfer between Alice and Bob. In the secure protocol, we need to streamline the sliding window comparison operation. Another potential direction is to perform an efficient maximum finding algorithm which determines an approximate maximum value by not making all the comparisons.

2.2.3 Keyword Spotting

Keyword spotting is a similar problem in which we try to find if one or more of a specified set of key words or phrases has occurred in a recording. The main difference of this problem from music matching is that the keywords can have different characteristics from the original recording as they can spoken at a different rate, tone, and even by a different speaker. Instead of representing the snippet by set of frames, a common approach consist of using an HMM for each keyword trained over recordings of the word. For a given target speech recording, Bob can find the probability of the HMM generating a set of frames efficiently using the forward algorithm. Given all such speech recordings in his collection, Bob needs to find the one having the set of frames which results in the maximum probability. Alice can also use a generic "garbage" HMM trained on all other speech is used in opposition to the model for the key words. Spotting is performed by comparing the likelihoods of the models for the keywords to that of the garbage model on the speech recording.

The above technique suffices in the case of keyword spotting with a single keyword. If Alice is using a key-phrase, she can model this as a concatenation of HMMs trained on individual keywords along with the HMM for the garbage model as shown in Figure 2.4. The garbage model is matched to all other words except those in the key-phrase. Similar to continuous speech recognition, keyword spotting proceeds by processing the given speech recording using the concatenated HMM and then identifying the most likely path using Viterbi decoding. By performing this processing on all the recordings in the dictionary, Bob can determine the position in which each word of the key-phrase occurs.

Privacy Issues

The main privacy requirement in keyword spotting is that Bob should not know which keywords are being used by Alice and also be able to communicate to her the positions in which the keywords occur oblivious to himself. The second requirement is important because if Bob learns the position of the keyword occurrence, he can try to identify the

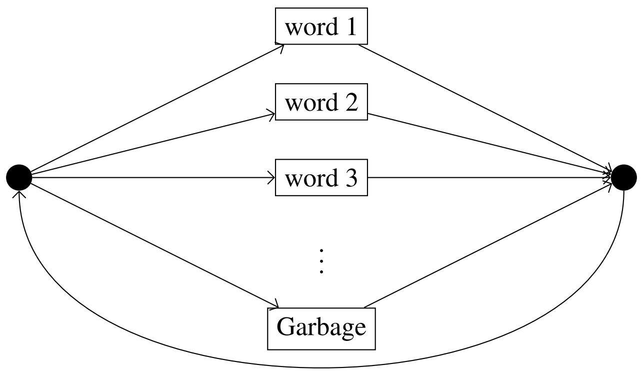

Figure 2.4: Concatenated HMMs for keyword spotting.

keyword from the original recording. For each HMM in the model, Alice and Bob need to perform Viterbi decoding securely, this makes privacy preserving keyword spotting very expensive in terms of computation and data transfer costs. The same ideas for privacy preserving music matching can be used to make keyword spotting more efficient.

2.2.4 Speech Recognition

Speech recognition is a type of pattern recognition problem, where the input is a stream of sampled and digitized speech data and the desired output is the sequence of words that were spoken. The pattern matching involves combining acoustic and language models to evaluate features which capture the spectral characteristics of the incoming speech signal. Most modern speech recognition systems use HMMs as the underlying model.

We view a speech signal as a sequence of piecewise stationary signals and an HMM forms a natural representation to output such a sequence of frames. We view the HMM to model the process underlying the observations as going through a number of states, in case of speech, the process can be thought of as the vocal tract of the speaker. We model each state of the HMM using a Gaussian mixture model (GMM) and correspondingly we can calculate the likelihood for an observed frame being generated by that model. The parameters of the HMM are trained over the labeled speech data using the Baum-Welch algorithm. We can use an HMM to model a phoneme which is the smallest segmental unit of sound which can provide meaningful distinction between two utterances. Alternatively, we can also use an HMM to model a complete word by itself or by concatenating the HMMs modeling the phonemes occurring in the word.

A useful way of visualizing speech recognition of an isolated word using an HMM is by a trellis shown in Figure 2.5. Every edge in the graph represents a valid transition in the HMM over a single time step and every node represents the event of a particular observation frame being generated from a particular state. The probability of an HMM generating a complete sequence of frames can be efficiently computed using the forward algorithm. Similarly, the state sequence having the maximum probability for an observation sequence can be found using the Viterbi algorithm. In order to perform isolated word recognition, we train an HMM model for each word in the dictionary as a template. Given a new utterance, we match it to each of the HMM model and choose the one with the highest probability as the recognized word.

However, we cannot directly use this approach of training multiple HMMs for continuous speech recognition as extremely large number of sentences are possible and there much greater amount of variation in speaking rate. This is circumvented by concatenating the HMMs trained over individual words as shown in Figure 2.6. We introduce optional silence HMMs to model silences between the words. The recognition is performed by running Viterbi decoding to find the most path of states through the composed HMM and thereby finding the most likely sequence of words.

However, such an HMM composition graph needs to be specified to the speech recognition system and connecting all the models manually becomes complicated for recognition of a non-trivial set of sentences. A common approach is to develop a probabilistic context-free grammar for our recognition task and automatically perform the graph composition using it. This is used in simple dialog systems and other command and control settings. To recognize uncon-

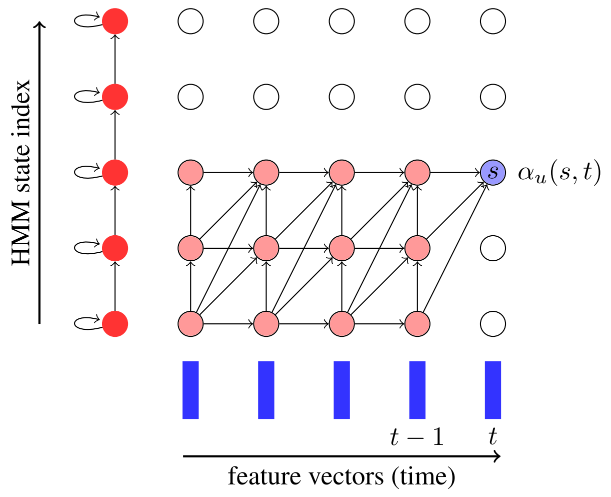

Figure 2.5: A trellis showing all possible paths of an HMM while recognizing a sequence of frames.

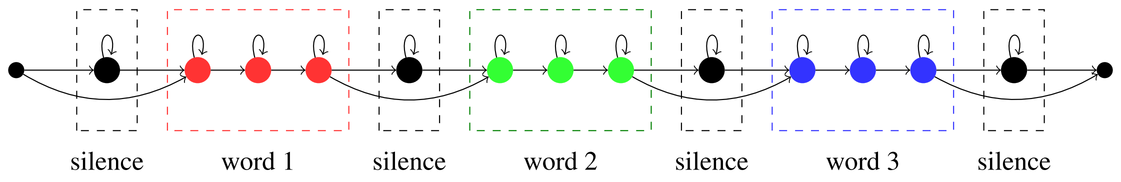

Figure 2.6: Concatenated HMMs for continuous speech recognition.

strained word sequences, state of the art large vocabulary speech recognition systems use n-gram language models where the probability of a word is assumed to be dependent only on the previous $n-1$ words. The n-gram models are learned from training data with the application of techniques such as smoothing and back-off to account for unseen n-grams. Using the Bayes rule, we model the posterior probability of a signal given the target word sequence.

$$\text{word}_1, \ldots, \text{word}_n = \underset{w_1,\ldots,w_n}{\text{argmax}} \underbrace{P(\text{signal}|w_1, \ldots, w_n)}_{textAcousticmodel(HMM)} \underbrace{P(w_1, \ldots, w_n)}_{\text{Language Model}}. \tag{2.1}$$

This allows us to compose the language graph containing individual word HMMs and the transition probabilities derived from the language model. In Figure 2.7, we show a compact representation of a trigram graph for two words.

Any HMM based recognition involves searching through such a graph structure. It should be noted that a vast majority of the paths will lead fail and the best path comparison can be made using only a small selection of such paths. This significant reduction in complexity and size of the model is achieved by pruning across all active states at any given time step. Using a state-level beam, we can prune poorly scoring states or alternatively using a word-level beam, we can prune poorly scoring words.

Privacy Issues

We consider speech recognition as a service where the server has a speech recognition model and the client has the audio data. For privacy preserving speech recognition, we require that the server performing the recognition does not observe the input audio data and conversely, the party possessing the input data does not observe the system parameters. The latter constraint is useful in settings where the server is interested in maintaining a pay per use system. Also, if the HMM might be trained over valuable and possibly private training data, analyzing the system parameters can potentially provide information about the training data. Analogously, training the HMM and language models needs to be performed without having access to the private speech data. This problem of privacy preserving HMM

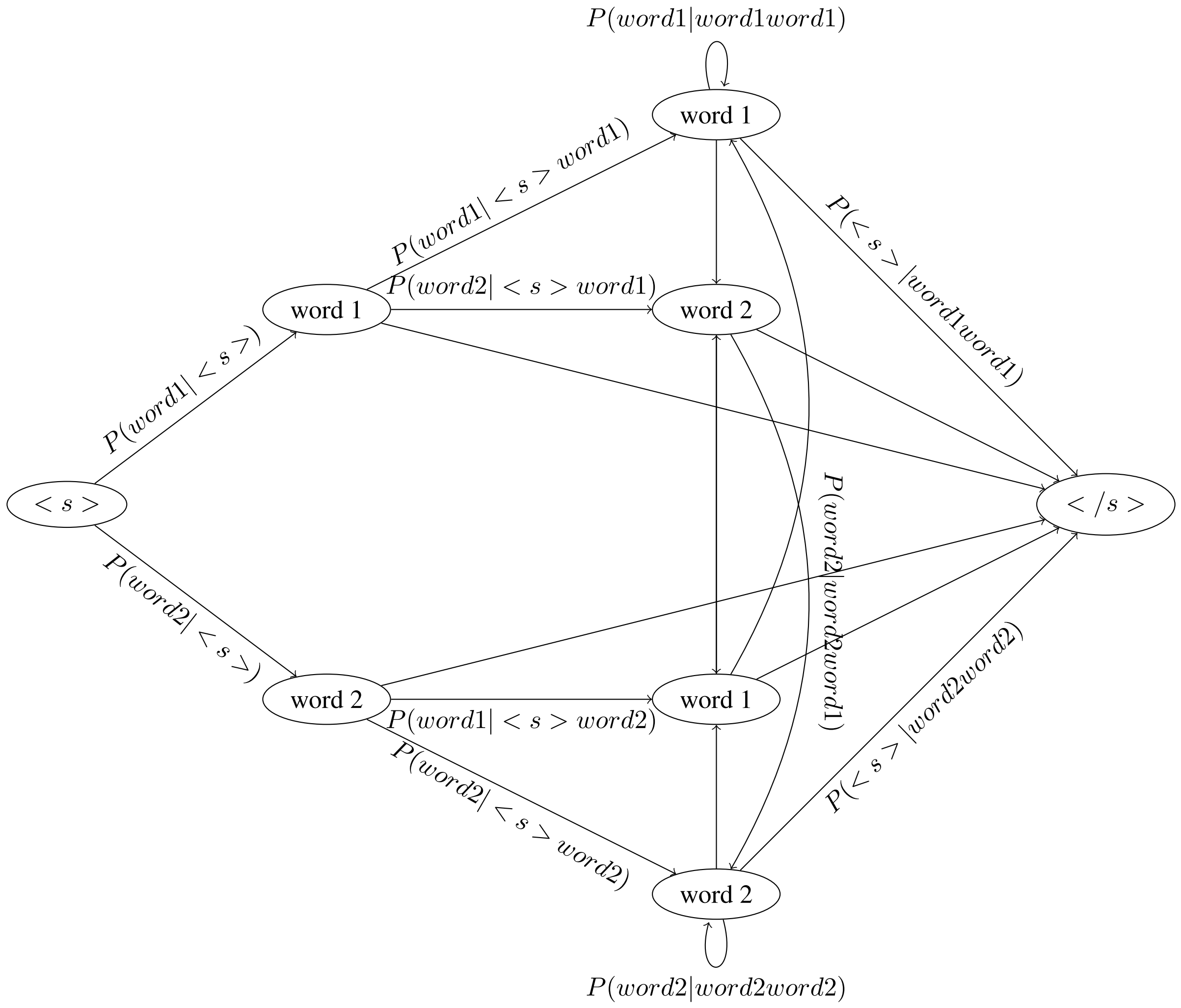

Figure 2.7: Trigram language graph for recognizing $< s >$,word1, word2,$< /s >$.

inference falls under the area of secure multiparty computation (Section 3.1), and suitable protocols can be designed using primitives such as homomorphic encryption. Isolated word recognition using a single HMM per word can be performed using the forward algorithm. It is relatively straightforward to construct a practical privacy preserving solution because of the linear complexity of the forward algorithm.

In continuous speech recognition by composite HMMs, the client and server need to collaborate in constructing a trellis structure indexed by the HMM states and the input sequences, and performing Viterbi decoding to identify the optimal path. While we need to match one HMM per word in case of isolated word recognition, the client privately needs to compare each frame of the input sequence to a graph of HMMs. This poses a number of problems owing to the size of the search space and the sheer number of operations that need to be performed privately. Pruning involves eliminating search paths in the graph which are unlikely to be the best paths. By evaluating individual search paths, the server will gain a lot of information about what words are likely or unlikely to be in the search path using the input data belonging to the user. Hence, by its nature, pruning is contrary to the privacy constraints and we cannot make advantage of it to increase the speed and scalability of the recognition.

Given the state of the art of research, we believe that continuous speech recognition with privacy constraints is currently infeasible.

Chapter 3

Background on Privacy and Security

3.1 Secure Multiparty Computation

Secure Multiparty Computation (SMC) consists of the setting in which individual parties jointly compute a function of their private input data with the constraint that no party should learn anything apart from the output and the intermediate steps of the computation. Informally, the security of an SMC protocol is defined as computational indistinguishability from an ideal functionality in which a trusted arbitrator accepts the input from all the parties, performs the computation and returns the results to all the parties. The actual protocol is secure if it does not leak any information beyond what is leaked in the ideal functionality. Please refer to [Beaver, 1991; Canetti, 2000; Goldreich, 2004] for formal definitons and theoretical models for analyzing SMC protocols.

Yao [1982] originally illustrated SMC using the millionaire problem where two parties are trying to identify who has more money without disclosing the actual values of their amounts. In this seminal work, Yao also showed that any polynomial-time multiparty computation can be performed with privacy in general but the generic techniques result in computationally impractical solutions. Recent research has focused on constructing efficient privacy preserving protocols for specific problems.

3.1.1 Data Setup and Privacy Conditions

We formally define the problem, in which multiple parties are interested in computing a function over their collectively held datasets without disclosing any information to each other. For simplicity, we first consider the case of two parties, Alice and Bob. The parties are assumed to be *semi-honest* which means that the parties will follow the steps of the protocol correctly and will not try to cheat by passing falsified data aimed at extracting information about other parties. The parties are assumed to be curious in the sense that they may record the outcomes of all intermediate steps of the protocol to extract any possible information.

We assume that both the datasets can be represented as matrices in which columns and rows correspond to the data samples and the features, respectively. For instance, the individual email collections of Alice and Bob are represented as matrices A and B respectively, in which the columns correspond to the emails, and the rows correspond to the words. The entries of these matrices represent the frequency of occurrence of a given word in a given email. The combined dataset may be split between Alice and Bob in two possible ways. In a *horizontal split* or data split, both Alice and Bob have a disjoint set of data samples with the same features. The aggregate dataset is obtained by concatenating columns given by the data matrix $M = \begin{bmatrix} A & B \end{bmatrix}$ and correlation matrix M^TM. In a *vertical split* or feature split, Alice and Bob have different features of the same data. The aggregate data matrix M is obtained by concatenating rows given by the data matrix $M = \begin{bmatrix} A \\ B \end{bmatrix}$ and correlation matrix MM^T.

3.1.2 Tools for Constructing SMC protocols

1. **Homomorphic Encryption.** A homomorphic encryption (HE) cryptosystem allows for operations to be perform on the encrypted data without requiring to know the unencrypted values. If $\cdot$ and $+$ are two operators and x and y are two plaintext elements, a homomorphic encryption function $\xi()$ satisfies $E[x] \cdot E[y] = E[x+y]$.

   Owing to this property, simple operations can be performed directly on the ciphertext, thereby resulting in some desirable manipulations on the underlying plaintext messages. Additively homomorphic cryptosystems have been proposed in the literature by Benaloh [1994], Paillier [1999] and Damgård and Jurik [2001]. The protocols that we construct in this paper can be executed with any of these cryptosystems. We primarily use the asymmetric key Paillier cryptosystem [Paillier, 1999] which provides additive homomorphism as well as semantic security.

   (a) **Key Generation.** Choose two large prime numbers p, q, and let $N = pq$. Denote by $\mathbb{Z}^*_{N^2} \subset \mathbb{Z}_{N^2} = \{0, 1, ..., N^2 - 1\}$ the set of non-negative integers that have multiplicative inverses modulo N^2. Select $g \in \mathbb{Z}^*_{N^2}$ such that $\gcd(L(g^\lambda \mod N^2), N) = 1$, where $\lambda = \text{lcm}(p-1, q-1)$, and $L(x) = \frac{x-1}{N}$. Let (N, g) be the public key, and (p, q) be the private key.

   (b) **Encryption.** Let $m \in \mathbb{Z}_N$ be a plaintext. Then, the ciphertext is given by

   $$\xi_r(m) = g^m \cdot r^N \mod N^2 \tag{3.1}$$

   where $r \in \mathbb{Z}^*_N$ is a number chosen at random.

   (c) **Decryption.** Let $c \in \mathbb{Z}_{N^2}$ be a ciphertext. Then, the corresponding plaintext is given by

   $$\psi(\xi_r(m)) = \frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} = m \mod N \tag{3.2}$$

   Note that decryption works irrespective of the value of r used during encryption. Since r can be chosen at random for every encryption, the Paillier cryptosystem is probabilistic, and therefore semantically secure. It can now be verified that the following homomorphic properties hold for the mapping (3.1) from the plaintext set $(\mathbb{Z}_N, +)$ to the ciphertext set $(\mathbb{Z}^*_{N^2}, \cdot)$,

   $$\psi(\xi_{r_1}(m_1)\xi_{r_2}(m_2) \mod N^2) = m_1 + m_2 \mod N$$
   $$\psi(\,[\,\xi_r(m_1)\,]^{m_2} \mod N^2) = m_1 m_2 \mod N$$

   where $r_1, r_2 \in \mathbb{Z}^*_N$ and $r_1 \neq r_2$ in general.

2. **Oblivious Transfer.**

   Oblivious transfer (OT) was originally introduced in Rabin [1981]. In an oblivious transfer protocol, the sender sends a collection of items to the receiver but remains oblivious about which item is valid.

   Suppose that Bob has n messages $m_1, m_2, \ldots m_n$ and Alice has the index $1 \leq i \leq n$, oblivious transfer ensures that (a) Alice obtains m_i but discovers nothing about the other messages (b) Bob does not discover i. There are many known ways to implement oblivious transfer such as those given by Naor and Pinkas [1999]. Also, it has been shown that oblivious transfer is a sufficient primitive, *i.e.*, it can be used for secure evaluation any function, provided that function can be represented as an algebraic circuit. However, evaluating general functions using oblivious transfer exclusively is very expensive in terms of data transfer and computational costs; we use oblivious transfer sparingly in our protocols.

3. **Blind and Permute.** If the input $c_1, c_2, \ldots, c_n$ is additively split among two parties with one party having $a_1, a_2, \ldots, a_n$ and other party having $b_1, b_2, \ldots, b_n$, the parties can participate in a blind and permute protocol [Atallah and Li, 2005] to obtain a randomly permuted additive split of the input such that the permutation is not known to either of the parties. This is useful in cases such as computing $\max c_i$ securely. The blind and permute primitive in turn uses homomorphic encryption and oblivious transfer.

3.1.3 Related Work on SMC Protocols for Machine Learning and Speech Processing

There has been a lot of work on constructing privacy preserving algorithms for specific problems in privacy preserving data mining such as decision trees [Vaidya et al., 2008a], set matching [Freedman et al., 2004], clustering [Lin et al., 2005], association rule mining [Kantarcioglu and Clifton, 2004], naive Bayes classification [Vaidya et al., 2008b], support vector machines [Vaidya et al., 2008c]. Please refer to [Lindell and Pinkas, 2009] for a summary of recent work.

However, there has been somewhat limited work on SMC protocols for speech processing applications, which is one of the motivations for this thesis. Probabilistic inference with privacy constraints is a relatively unexplored area of research. The only detailed treatment of privacy-preserving probabilistic classification appears in [Smaragdis and Shashanka, 2007]. In that work, inference via HMMs is performed on speech data using existing cryptographic primitives. The protocols are based on repeated invocations of privacy-preserving two-party maximization algorithms, in which both parties incur exactly the same protocol overhead.

3.2 Differential Privacy

The differential privacy model was originally introduced by Dwork [2006]. Given any two databases D and D' differing by one element, which we will refer to as *adjacent databases*, a randomized query function M is said to be differentially private if the probability that M produces a response S on D is close to the probability that M produces the same response S on D'. As the query output is almost the same in the presence or absence of an individual entry with high probability, nothing can be learned about any individual entry from the output. Differential privacy is formally defined as follows.

Definition. A randomized function M with a well-defined probability density P satisfies ϵ-differential privacy if, for all adjacent databases D and D' and for any set of outcomes $S \in range(M)$,

$$\left| \log \frac{P\left[M(D) \in S\right]}{P\left[M(D') \in S\right]} \right| \leq \epsilon. \tag{3.3}$$

Dwork et al. [2006] proposed the *exponential mechanism* for creating functions satisfying ϵ-differential privacy by adding a perturbation term from the Laplace distribution scaled by the function *sensitivity* and ϵ, which is defined as the maximum change in the value of the function when evaluated over adjacent datasets.

In a classification setting, the training dataset may be thought of as the database and the algorithm learning the classification rule as the query mechanism. A classifier satisfying differential privacy implies that no additional details about the individual training data instances can be obtained with certainty from output of the learning algorithm, beyond the *a priori* background knowledge. An adversary who observes the values for all except one entry in the dataset and has prior information about the last entry cannot learn anything with high certainty about the value of the last entry beyond what was known *a priori* by observing the output of the mechanism over the dataset.

Differential privacy provides an *ad omnia* guarantee as opposed to most other models that provide *ad hoc* guarantees against a specific set of attacks and adversarial behaviors. By evaluating the differentially private classifier over a large number of test instances, an adversary cannot learn the exact form of the training data. This is a stronger guarantee than SMC where the only condition is that the parties are not able to learn anything about the individual data beyond what may be inferred from the final result of the computation. For instance, when the outcome of the computation is a classifier, it does not prevent an adversary from postulating the presence of data instances whose absence might change the decision boundary of the classifier, and verifying the hypothesis using auxiliary information if any. Moreover, for all but the simplest computational problems, SMC protocols tend to be highly expensive, requiring iterated encryption and decryption and repeated communication of encrypted partial results between participating parties.

3.2.1 Related Work on Differentially Private Machine Learning

The earlier work on differential privacy was related to functional approximations for simple data mining tasks and data release mechanisms [Dinur and Nissim, 2003; Dwork and Nissim, 2004; Blum et al., 2005; Barak et al., 2007].

Although many of these works have connection to machine learning problems, more recently the design and analysis of machine learning algorithms satisfying differential privacy has been actively studied. Kasiviswanathan et al. [2008] present a framework for converting a general agnostic PAC learning algorithm to an algorithm that satisfies privacy constraints. Chaudhuri and Monteleoni [2008] use the exponential mechanism [Dwork et al., 2006] to create a differentially private logistic regression classifier by adding Laplace noise to the estimated parameters. They propose another differentially private formulation which involves modifying the objective function of the logistic regression classifier by adding a linear term scaled by Laplace noise. The second formulation is advantageous because it is independent of the classifier sensitivity which difficult to compute in general and it can be shown that using a perturbed objective function introduces a lower error as compared to the exponential mechanism.

However, the above mentioned differentially private classification algorithms only address the problem of binary classification. Although it is possible to extend binary classification algorithms to multi-class using techniques like one-vs-all, it is much more expensive to do so as compared to a naturally multi-class classification algorithm. Jagannathan et al. [2009] present a differentially private random decision tree learning algorithm which can be applied to multi-class classification. Their approach involves perturbing leaf nodes using the sensitivity method, and they do not provide theoretical analysis of excess risk of the perturbed classifier.

Chapter 4

Preliminary Work

We review the preliminary results of our research.

4.1 Techniques

4.1.1 Privacy Preserving Eigenvector Computation

We developed a protocol for computing the principal eigenvector of the combined data shared by multiple parties coordinated by a semi-honest arbitrator Trent [Pathak and Raj, 2010c]. This protocol is significantly more efficient than the previous approaches towards this problem which are based on expensive QR decomposition of correlation matrices. This was our first attempt at designing privacy preserving protocols. While it is not directly related to speech processing per se, many algorithms for probabilistic inference such as the forward backward algorithm for HMMs perform computation which is similar to eigenvector computation.

The power iteration algorithm computes the principal eigenvector of $M^T M$ by updating and normalizing the vector x_t until convergence. Starting with a random vector x_0, we calculate

$$x_{i+1} = \frac{M^T M\, x_i}{\|M^T M\, x_i\|}.$$

We first consider the case in which two parties Alice and Bob with horizontally split data matrices $A_{k \times m}$ and $B_{k \times n}$ compute the principal eigenvector of the combined data in collaboration with an untrusted third party arbitrator Trent. For privacy, we split the vector x_i into two parts, α_i and β_i. α_i corresponds to the first m components of x_i and β_i corresponds to the remaining n components. In each iteration, we need to securely compute

$$M^T M x_i = \begin{bmatrix} A^T A & A^T B \\ B^T A & B^T B \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} A^T (A\alpha_i + B\beta_i) \\ B^T (A\alpha_i + B\beta_i) \end{bmatrix} = \begin{bmatrix} A^T u_i \\ B^T u_i \end{bmatrix} \tag{4.1}$$

where $u_i = A\alpha_i + B\beta_i$. After convergence, α_i and β_i will represent shares held by Alice and Bob of the principal eigenvector of $M^T M$.

An iteration of the algorithm proceeds as illustrated in Figure 4.1. At the outset Alice and Bob randomly generate component vectors α_0 and β_0 respectively. At the beginning of the i^{th} iteration, Alice and Bob possess component vectors α_i and β_i respectively. They compute the product of their data and their corresponding component vectors as $A\alpha_i$ and $B\beta_i$. To compute u_i, Alice and Bob individually transfer these products to Trent. Trent adds the contributions from Alice and Bob by computing

$$u_i = A\alpha_i + B\beta_i.$$

He then transfers u_i back to Alice and Bob, who then individually compute $A^T u_i$ and $B^T u_i$, without requiring data from one other. For normalization, Alice and Bob also need to securely compute the term

$$\|M^T M\, x_i\| = \sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}. \tag{4.2}$$

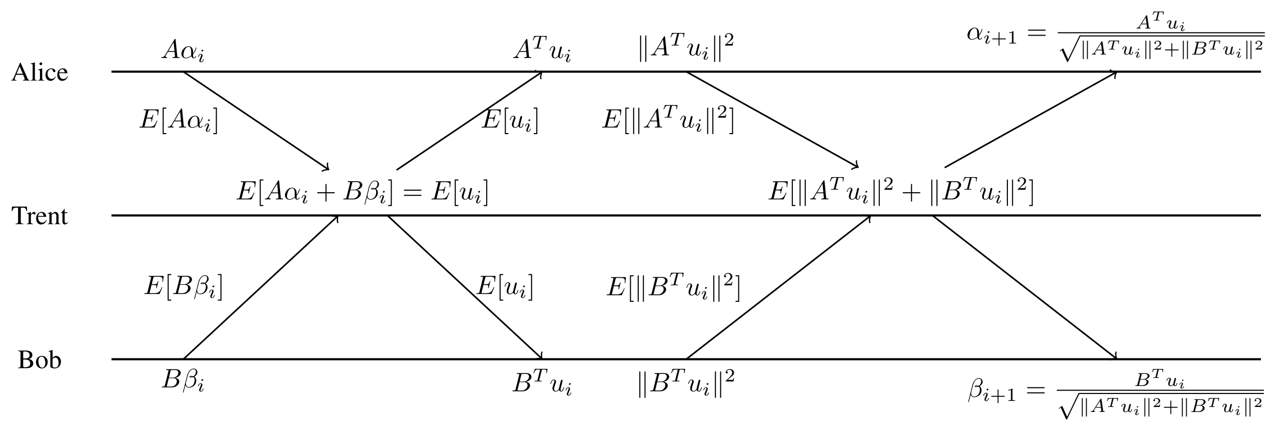
Alice
Trent
Bob
$A\alpha_i$
$A^T u_i$
$\|A^T u_i\|^2$
$\alpha_{i+1} = \frac{A^T u_i}{\sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}}$
$E[A\alpha_i]$
$E[u_i]$
$E[\|A^T u_i\|^2]$
$E[A\alpha_i + B\beta_i] = E[u_i]$
$E[\|A^T u_i\|^2 + \|B^T u_i\|^2]$
$E[B\beta_i]$
$E[u_i]$
$E[\|B^T u_i\|^2]$
$B\beta_i$
$B^T u_i$
$\|B^T u_i\|^2$
$\beta_{i+1} = \frac{B^T u_i}{\sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}}$

Figure 4.1: Visual description of the protocol.

Again, Alice and Bob compute the individual terms $\|A^T u_i\|^2$ and $\|B^T u_i\|^2$ respectively and transfer it to Trent. As earlier, Trent computes the sum

$$\|A^T u_i\|^2 + \|B^T u_i\|^2$$

and transfers it back to Alice and Bob. Finally, Alice and Bob respectively update α and β vectors as

$$\begin{aligned} u_i &= A\alpha_i + B\beta_i, \\ \alpha_{i+1} &= \frac{A^T u_i}{\sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}}, \\ \beta_{i+1} &= \frac{B^T u_i}{\sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}}. \end{aligned} \tag{4.3}$$

The algorithm terminates when the α and β vectors converge.

The basic protocol described above is provably correct. After convergence, Alice and Bob end up with the principal eigenvector of the row space of the combined data, as well as concatenative shares of the column space which Trent can gather to compute the principal eigenvector. However the protocol is not completely secure; Alice and Bob obtain sufficient information about properties of each others' data matrices, such as their column spaces, null spaces, and correlation matrices. We present a series of modifications to the basic protocol so that such information is not revealed.

Homomorphic Encryption: Securing the data from Trent

The central objective of the protocol is to prevent Trent from learning anything about either the individual data sets or the combined data other than the principal eigenvector of the combined data. Trent receives a series of partial results of the form $AA^T u$, $BB^T u$ and $MM^T u$. By analyzing these results, he can potentially determine the entire column spaces of Alice and Bob as well as the combined data. To prevent this, we employ an additive homomorphic cryptosystem.

At the beginning of the protocol, Alice and Bob obtain a shared public key/private key pair for an additive homomorphic cryptosystem from an authenticating authority. The public key is also known to Trent who, however, does not know the private key; While he can encrypt data, he cannot decrypt it. Alice and Bob encrypt all transmissions to Trent, at the first transmission step of each iteration Trent receives the encrypted inputs $E[A\alpha_i]$ and $E[B\beta_i]$. He multiplies the two element by element to compute $E[A\alpha_i] \cdot E[B\beta_i] = E[A\alpha_i + B\beta_i] = E[u_i]$. He returns $E[u_i]$ to both Alice and Bob who decrypt it with their private key to obtain u_i. In the second transmission step of each iteration, Alice and Bob send $E[\|A^T u_i\|^2]$ and $E[\|B^T u_i\|^2]$ respectively to Trent, who computes the encrypted sum

$$E\left[\|A^T u_i\|^2\right] \cdot E\left[\|B^T u_i\|^2\right] = E\left[\|A^T u_i\|^2 + \|B^T u_i\|^2\right]$$

and transfers it back to Alice and Bob, who then decrypt it to obtain $\|A^T u_i\|^2 + \|B^T u_i\|^2$, which is required for normalization.

This modification does not change the actual computation of the power iterations in any manner. Thus the procedure remains as correct as before, except that Trent now no longer has any access to any of the intermediate computations. At the termination of the algorithm he can now receive the converged values of α and β from Alice and Bob, who will send it in clear text.

Random Scaling: Securing the Column Spaces

After Alice and Bob receive $u_i = A\alpha_i + B\beta_i$ from Trent, Alice can calculate $u_i - A\alpha_i = B\beta_i$ and Bob can calculate $u_i - B\beta_i = A\alpha_i$. After a sufficient number of iterations, particularly in the early stages of the computation (when u_i has not yet converged) Alice can find the column space of B and Bob can find the column space of A. Similarly, by subtracting their share from the normalization term returned by Trent, Alice and Bob are able to find $\|B^T u_i\|^2$ and $\|A^T u_i\|^2$ respectively.

In order to prevent this, Trent multiplies u_i with a randomly generated scaling term r_i that he does not share with anyone. Trent computes

$$(E[A\alpha_i] \cdot E[B\beta_i])^{r_i} = E[r_i(A\alpha_i + B\beta_i)] = E[r_i u_i]$$

by performing element-wise exponentiation of the encrypted vector by r_i and transfers $r_i u_i$ to Alice and Bob. By using a different value of r_i at each iteration, Trent ensures that Alice and Bob are not able to calculate $B\beta_i$ and $A\alpha_i$ respectively. In the second step, Trent scales the normalization constant by r_i^2,

$$\left(E\left[\|A^T u_i\|^2\right] \cdot E\left[\|B^T u_i\|^2\right]\right)^{r_i^2} = E\left[r_i^2\left(\|A_i^T u\|^2 + \|B_i^T u\|^2\right)\right].$$

Normalization causes the r_i factor to cancel out and the update rules remain unchanged.

$$\begin{aligned} u_i &= A\alpha_i + B\beta_i, \\ \alpha_{i+1} &= \frac{r_i A^T u_i}{\sqrt{r_i^2\left(\|A^T u_i\|^2 + \|B^T u_i\|^2\right)}} = \frac{A^T u_i}{\sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}}, \\ \beta_{i+1} &= \frac{r_i B^T u_i}{\sqrt{r_i^2\left(\|A^T u_i\|^2 + \|B^T u_i\|^2\right)}} = \frac{B^T u_i}{\sqrt{\|A^T u_i\|^2 + \|B^T u_i\|^2}}. \end{aligned} \tag{4.4}$$

The random scaling does not affect the final outcome of the computation, and the algorithm remains correct as before.

Data Padding: Securing null spaces

In each iteration, Alice observes one vector $r_i u_i = r_i(A\alpha_i + B\beta_i)$ in the column space of $M = [A\ B]$. Alice can calculate the *null space* $H(A)$ of A, given by

$$H(A) = \{x \in \mathbb{R}^m | Ax = 0\}$$

and pre-multiply a non-zero vector $x \in H(A)$ with $r_i u_i$ to calculate

$$x r_i u_i = r_i x(A\alpha_i + B\beta_i) = r_i x B\beta_i.$$

This is a projection of $B\beta_i$, a vector in the column space of B into the null space $H(A)$. Similarly, Bob can find projections of $A\alpha_i$ in the null space $H(B)$. While considering the projected vectors separately will not give away much information, after several iterations Alice will have a projection of the column space of B on the null space of A, thereby learning about the component's of Bob's data that lie in her null space. Bob can similarly learn about the component's of Alice's data that lie in his null space.

In order to prevent this, Alice *pads* her data matrix A by concatenating it with a random matrix $P_a = r_a I_{k\times k}$, to obtain $\begin{bmatrix} A & P_a \end{bmatrix}$ where r_a is a positive scalar chosen by Alice. Similarly, Bob pads his data matrix B with $P_b = r_b I_{k\times k}$

to obtain $\begin{bmatrix} B & P_b \end{bmatrix}$ where r_b is a different positive scalar chosen by Bob. This has the effect of hiding the null spaces in both their data sets. In the following theorem, we prove that the eigenvectors of the combined data do not change after padding, while every eigenvalue λ of MM^T is now modified to $\lambda + r_a + r_b$. Please refer to appendix for the proof.

Theorem 4.1.1. *Let $\bar{M} = \begin{bmatrix} M & P \end{bmatrix}$ where M is a $s \times t$ matrix, and P is a $s \times s$ orthogonal matrix. If $\bar{v} = \begin{bmatrix} v_{t\times 1} \\ v'_{s\times 1} \end{bmatrix}$ is an eigenvector of $\bar{M}^T \bar{M}$ corresponding to an eigenvalue λ, then v is an eigenvector of $M^T M$.*

While the random factors r_a and r_b prevent Alice and Bob from estimating the eigenvalues of the data, the computation of principal eigenvector remains correct as before.

Oblivious Transfer: Securing Krylov spaces

For a constant c, we can show that the vector $u_i = A\alpha_i + B\beta_i$ is equal to $cMM^T u_{i-1}$. The sequence of vectors $U = \{u_1, u_2, u_3, \ldots\}$ form the Krylov subspace $(MM^T)^n u_1$ of the matrix MM^T. Knowledge of this series of vectors can reveal all eigenvectors of MM^T. Consider $u_0 = c_1 v_1 + c_2 v_2 + \cdots$, where v_i is the i^{th} eigenvector. If λ_j is the j^{th} eigenvalue, we have $u_i = c_1\lambda_1 v_1 + c_2\lambda_2 v_2 + \cdots$. We assume wlog that the eigenvalues λ are in a descending order, *i.e.*, $\lambda_j \geq \lambda_k$ for $j < k$. Let u_{conv} be the normalized converged value of u_i which is equal to the normalized principal eigenvector v_1.

Let $w_i = u_i - (u_i \cdot u_{conv})u_i$ which can be shown to be equal to $c_2\lambda_2 v_2 + c_3\lambda_3 v_3 + \cdots$, *i.e.*, a vector with no component along v_1. If we perform power iterations with initial vector w_1, the converged vector w_{conv} will be equal to the eigenvector corresponding to the second largest eigenvalue. Hence, once Alice has the converged value, u_{conv}, she can subtract it out of all the stored u_i values and determine the second principal eigenvector of MM^T. She can repeat the process iteratively to obtain all eigenvectors of MM^T, although in practice the estimates become noisy very quickly. The following modification prevents Alice and Bob from identifying the Krylov space with any certainty and they are thereby unable to compute the additional eigenvectors of the combined data.

We introduce a form of oblivious transfer (OT) [Rabin, 1981] in the protocol. We assume that Trent stores the encrypted results of intermediate steps at every iteration. After computing $E[r_i u_i]$, Trent either sends this quantity to Alice and Bob with a probability p or sends a random vector $E[u'_i]$ of the same size ($k \times 1$) with probability $1 - p$. As the encryption key of the cryptosystem is publicly known, Trent can encrypt the vector u'_i. Alice and Bob do not know whether they are receiving $r_i u_i$ or u'_i. If a random vector is sent, Trent continues with the protocol, but ignores the terms Alice and Bob return in the next iteration, $E[A\alpha_{i+1}]$ and $E[B\beta_{i+1}]$. Instead, he sends the result of a the last non-random iteration j, $E[r_j u_j]$, thereby restarting that iteration.

This sequence of data sent by Trent is an example of a Bernoulli Process [Papoulis, 1984]. An illustrative example of the protocol is shown in Figure 4.2. In the first two iterations, Trent sends valid vectors $r_1 u_1$ and $r_2 u_2$ back to Alice and Bob. In the beginning of the third iteration, Trent receives and computes $E[r_3 u_3]$ but sends a random vector u'_3. He ignores what Alice and Bob send him in the fourth iteration and sends back $E[r_3 u_3]$ instead. Trent then stores the vector $E[r_4 u_4]$ sent by Alice and Bob in the fifth iteration and sends a random vector u'_2. Similarly, he ignores the computed vector of the sixth iteration and sends u'_3. Finally, he ignores the computed vector of the seventh iteration and sends $E[r_4 u_4]$.

$$\begin{tikzcd}
r_1u_1 \arrow[r] & r_2u_2 \arrow[r] \arrow[rr, bend left] & u'_1 \arrow[r] & r_3u_3 \arrow[r] \arrow[rrr, bend right] & u'_2 \arrow[r] & u'_3 \arrow[r] & r_4u_4
\end{tikzcd}$$

Figure 4.2: An example of the protocol execution with oblivious transfer.

This modification has two effects – firstly it prevents Alice and Bob from identifying the Krylov space with certainty. As a result, they are now unable to obtain additional Eigenvectors from the data. Secondly, oblivious transfer essentially obfuscates the projection of the column space of B on to the null space of A for Alice, and analogously for Bob by introducing random vectors. As Alice and Bob do not know which vectors are random, they cannot completely calculate the true projection of each others data on the null spaces. This is rendered less important if Alice and Bob pad their data as suggested in the previous subsection.

Alice and Bob can store the vectors they receive from Trent in each iteration. By analyzing the distribution of the normalized vectors, Alice and Bob can identify the random vectors using a simple outlier detection technique. To prevent this, one possible solution is for Trent to pick a previously computed value of $r_j u_j$ and add zero mean noise e_i, for instance, sampled from the Gaussian distribution.

$$u_i' = r_j u_j + e_i, \;\; e_i \sim \mathcal{N}(0, \sigma^2).$$

Instead of transmitting a perturbation of a previous vector, Trent can also use perturbed mean of a few previous $r_j u_j$ with noise. Doing this will create a random vector with the same distributional properties as the real vectors. The noise variance parameter σ controls the error in identifying the random vector from the valid vectors and how much error do we want to introduce in the projected column space.

Oblivious transfer has the effect of increasing the total computation as every iteration in which Trent sends a random vector is wasted. In any secure multi-party computation, there is an inherent trade-off between computation time and the degree of security. The parameter p which is the probability of Trent sending a non-random vector allows us to control this at a fine level based on the application requirements. As before, introducing oblivious transfer does not affect the correctness of the computation – it does not modify the values of the non-random vectors u_i.

Extension to Multiple Parties

As we mentioned before, the protocol can be naturally extended to multiple parties. Let us consider the case of N parties: $P_1, \ldots, P_N$ each having data $A_1, \ldots, A_N$ of sizes $k \times n_1, \ldots, k \times n_N$ respectively. The parties are interested in computing the principal eigenvector of the combined data without disclosing anything about their data. We make the same assumption about the parties and the arbitrator Trent being *semi-honest*. All the parties except Trent share the decryption key to the additive homomorphic encryption scheme and the encryption key is public.

In case of a data split, for the combined data matrix $M = \begin{bmatrix} A_1 & A_2 & \cdots & A_N \end{bmatrix}$, the correlation matrix is

$$M^T M = \begin{bmatrix} A_1^T A_1 & \cdots & A_1^T A_N \\ \vdots & \ddots & \vdots \\ A_N^T A_1 & \cdots & A_N^T A_N \end{bmatrix}.$$

We split the eigenvector into N parts, $\alpha_1, \ldots, \alpha_N$ of size $n_1, \ldots, n_N$ respectively, each corresponding to one party. For simplicity, we describe the basic protocol with homomorphic encryption; randomization and oblivious transfer can be easily added by making the same modifications as we saw in Sections 3.3. One iteration of the protocol starts with the i^{th} party computing $A_i\alpha_i$ and transferring to Trent the encrypted vector $E[A_i\alpha_i]$. Trent receives this from each party and computes

$$\prod_i E\left[A_i\alpha_i\right] = E\left[\sum_i A_i\alpha_i\right] = E[u]$$

where $u = \sum_i A_i\alpha_i$, and product is an element-wise operation. Trent sends the encrypted vector $E[u]$ back to $P_1, \ldots, P_N$ who decrypt it and individually compute $A_i^T u$. The parties individually compute $\|A_i^T u\|^2$ and send its encrypted value to Trent. Trent receives N encrypted scalars $E\left[\|A_i^T u\|^2\right]$ and calculates the normalization term

$$\prod_i E\left[\|A_i^T u\|^2\right] = E\left[\sum_i \|A_i^T u\|^2\right]$$

and sends it back to the parties. At the end of the iteration, the party P_i updates α_i as

$$u = \sum_i A_i \alpha_i^{(old)},$$

$$\alpha_i^{(new)} = \frac{A_i^T u}{\sqrt{\sum_i \|A_i^T u\|^2}}. \tag{4.5}$$

The algorithm terminates when any one party P_i converges on α_i.

4.1.2 Differentially Private Large Margin Gaussian Mixture Models

We developed a learning algorithm for discriminatively trained Gaussian mixture models which satisfies differential privacy [Pathak and Raj, 2010a,b] and is the first multiclass classification algorithm doing so.

Large Margin Gaussian Classifiers

There has been work on learning GMM while satisfying the large margin property [Sha and Saul, 2006]. This leads to the classification algorithm being robust to outliers with provably strong generalization guarantees.

We first consider the setting where each class is modeled as a single Gaussian ellipsoid. Each class is modeled by a set of K Gaussian ellipsoids. The decision rule is to assign an instance $\vec{x}_i$ to the class having smallest Mahalanobis distance [Mahalanobis, 1936] from $\vec{x}_i$ to the centroid of that class. We apply the large margin intuition about the classifier maximizing the distance of training data instances from the decision boundaries having a lower error. Formally, we require that for each training data instance $\vec{x}_i$ with label y_i, the distance from $\vec{x}_i$ to the centroid of class y_i is at least less than its distance from centroids of all other classes by one.

$$\forall c \neq y_i : \vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i \ \geq \ 1 + \vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i. \tag{4.6}$$

Analogous to support vector machines, the training algorithm is an optimization problem minimizing the *hinge loss* denoted by $[f]_+ = \max(0, f)$, with a linear penalty for incorrect classification. We use the sum of traces of inverse covariance matrices for each classes as a *regularization* term.

$$J(\mathbf{\Phi}, \vec{x}, \vec{y}) = \sum_i \sum_{c \neq y_i} \left[1 + \vec{x}_i^T (\mathbf{\Phi}_{y_i} - \mathbf{\Phi}_c) \vec{x}_i\right]_+ + \lambda \sum_c \text{trace}(\mathbf{\Psi}_c). \tag{4.7}$$

Generalizing to Mixtures of Gaussians

We extend the above classification framework to modeling each class as a mixture of K Gaussians ellipsoids. A simple extension is to consider each data instance $\vec{x}_i$ as having a mixture component m_i along with the label y_i. The mixture labels are not available *a priori*, these can be generated by training a generative GMM using the data instances in each class and selecting the mixture component with the highest posterior probability. Similar to the single Gaussian per class criterion, we require that for each training data instance $\vec{x}_i$ with label y_i and mixture component m_i, the distance from $\vec{x}_i$ to the centroid of the Gaussian ellipsoid for the mixture component m_i of label y_i is at least one greater than the minimum distance from $\vec{x}_i$ to the centroid of any mixture component of any other class.

$$\forall c \neq y_i : \min_m \vec{x}_i^T \mathbf{\Phi}_{cm} \vec{x}_i \ \geq \ 1 + \vec{x}_i^T \mathbf{\Phi}_{y_i, m_i} \vec{x}_i.$$

In order to maintain the convexity of the objective function, we use the property $\min_m a_m \geq -\log \sum_m e^{-a_m}$ to rewrite the above constraint as

$$\forall c \neq y_i : -\log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_{cm} \vec{x}_i} \ \geq \ 1 + \vec{x}_i^T \mathbf{\Phi}_{y_i, m_i} \vec{x}_i. \tag{4.8}$$

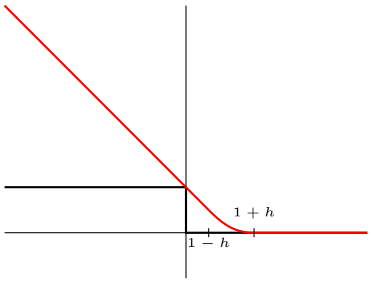

Figure 4.3: Hinge loss

As before, we minimize the hinge loss of misclassification along with the regularization term. The objective function becomes

$$J(\mathbf{\Phi}, \vec{x}, \vec{y}) = \sum_i \sum_{c \neq y_i} \left[1 + \vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i + \log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i} \right]_+ + \lambda \sum_{cm} \text{trace}(\mathbf{I_\Phi} \mathbf{\Phi}_{cm} \mathbf{I_\Phi}). \tag{4.9}$$

It can be observed that after this modification, the optimization problem remains a convex semidefinite program and is tractable to solve.

Making the Objective Function Differentiable

The hinge loss being non-differentiable is not convenient for our analysis; we replace it with a surrogate loss function called Huber loss l_h [Chapelle, 2007] which has similar characteristics as the hinge loss for small values of h. Let us denote $\vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i + \log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i}$ by $M(x_i, \mathbf{\Phi}_c)$ for conciseness. The Huber loss ℓ_h computed over data instances $(\vec{x}_i, y_i)$ becomes

$$\ell_h(\mathbf{\Phi}_c, \vec{x}_i, y_i) = \begin{cases} 0 & \text{if } M(x_i, \mathbf{\Phi}_c) > h, \\ \frac{1}{4h} \left[h - \vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i - \log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i} \right]^2 & \text{if } |M(x_i, \mathbf{\Phi}_c)| \leq h \\ -\vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i - \log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i} & \text{if } M(x_i, \mathbf{\Phi}_c) < -h. \end{cases} \tag{4.10}$$

Finally, the regularized Huber loss computed over the the training dataset $(\mathbf{x}, \vec{y})$ is given by

$$\begin{aligned} J(\mathbf{\Phi}, \vec{x}, \vec{y}) &= \sum_i \sum_{c \neq y_i} \ell_h \left[1 + \vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i + \log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i} \right] + \lambda \sum_{cm} \text{trace}(\mathbf{I_\Phi} \mathbf{\Phi}_{cm} \mathbf{I_\Phi}) \\ &= L(\mathbf{\Phi}, \vec{x}, \vec{y}) + N(\mathbf{\Phi}), \end{aligned} \tag{4.11}$$

where $L(\mathbf{\Phi}, \vec{x}, \vec{y})$ is the loss function and $N(\mathbf{\Phi})$ is the regularization term.

Differentially Private Formulation

We modify the large margin Gaussian mixture model formulation to satisfy differential privacy by introducing a perturbation term in the objective function. We generate the size $(d+1) \times (d+1)$ perturbation matrix $\mathbf{b}$ with density

$$P(\mathbf{b}) \propto \exp\left(-\frac{\epsilon}{2} \|\mathbf{b}\| \right), \tag{4.12}$$

where $\| \cdot \|$ is the Frobenius norm (element-wise ℓ_2 norm) and ϵ is the privacy parameter. One method of generating such a $\mathbf{b}$ matrix is to sample the norm $\|\mathbf{b}\|$ from $\Gamma\left((d+1)^2, \frac{2}{\epsilon}\right)$ and the direction of $\mathbf{b}$ at random.

Our proposed learning algorithm minimizes the following objective function $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$

$$J_p(\mathbf{\Phi}, \vec{x}, \vec{y}) = L(\mathbf{\Phi}, \vec{x}, \vec{y}) + N(\mathbf{\Phi}) + \sum_c \sum_{ij} b_{ij} \Phi_{cij}. \tag{4.13}$$

As the dimensionality of the perturbation matrix $\mathbf{b}$ is same as that of the classifier parameters $\mathbf{\Phi}_c$, the parameter space of $\mathbf{\Phi}$ does not change after perturbation. In other words, given two datasets $(\vec{x}, \vec{y})$ and $(\vec{x}', \vec{y}')$, if $\mathbf{\Phi}^{\mathbf{p}}$ minimizes $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$, it is always possible to have $\mathbf{\Phi}^{\mathbf{p}}$ minimize $J_p(\mathbf{\Phi}, \vec{x}', \vec{y}')$. This is a necessary condition for the classifier $\mathbf{\Phi}^{\mathbf{p}}$ satisfying differential privacy.

As the perturbation term is convex and positive semidefinite, the perturbed objective function $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ has the same properties as the unperturbed objective function $J(\mathbf{\Phi}, \vec{x}, \vec{y})$.

Proof of Differential Privacy

We prove that the classifier minimizing the perturbed optimization function $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ satisfies ϵ-differential privacy in the following theorem. Given a dataset $(\vec{x}, \vec{y}) = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_{n-1}, y_{n-1}), (\vec{x}_n, y_n)\}$, the probability of learning the classifier $\mathbf{\Phi}^p$ is close to the the probability of learning the same classifier $\mathbf{\Phi}^p$ given an adjacent dataset $(\vec{x}', \vec{y}') = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_{n-1}, y_{n-1}), (\vec{x}'_n, y'_n)\}$ differing wlog on the n^{th} instance. As we mentioned in the previous section, it is always possible to find such a classifier $\mathbf{\Phi}^p$ minimizing both $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ and $J_p(\mathbf{\Phi}, \vec{x}', \vec{y}')$ due to the perturbation matrix being in the same space as the optimization parameters.

Our proof requires a strictly convex perturbed objective function resulting in a unique solution $\mathbf{\Phi}^p$ minimizing it. This in turn requires that the loss function $L(\mathbf{\Phi}, \vec{x}, y)$ is strictly convex and differentiable, and the regularization term $N(\mathbf{\Phi})$ is convex. These seemingly strong constraints are satisfied by many commonly used classification algorithms such as logistic regression, support vector machines, and our general perturbation technique can be extended to those algorithms. In our proposed algorithm, the Huber loss is by definition a differentiable function and the trace regularization term is convex and differentiable. Additionally, we require that the difference in the gradients of $L(\mathbf{\Phi}, \vec{x}, y)$ calculated over for two adjacent training datasets is bounded. We prove this property in Lemma A.2.1.

Theorem 4.1.2. *For any two adjacent training datasets $(\vec{x}, \vec{y})$ and $(\vec{x}', \vec{y}')$, the classifier $\mathbf{\Phi}^p$ minimizing the perturbed objective function $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ satisfies differential privacy.*

$$\left| \log \frac{P(\mathbf{\Phi}^p | \vec{x}, \vec{y})}{P(\mathbf{\Phi}^p | \vec{x}', \vec{y}')} \right| \leq \epsilon',$$

where $\epsilon' = \epsilon + k$ for a constant factor $k = \log\left(1 + \frac{2\alpha}{n\lambda} + \frac{\alpha^2}{n^2\lambda^2}\right)$ with a constant value of α.

Analysis of Excess Risk

The perturbation introduced in the objective function comes at a cost of accuracy; when applied to any training dataset, the differentially private classifier will have an extra error as compared to the original classifier. We establish a bound on this excess risk.

In the remainder of this section, we denote the terms $J(\mathbf{\Phi}, \mathbf{x}, \mathbf{y})$ and $L(\mathbf{\Phi}, \mathbf{x}, \mathbf{y})$ by $J(\mathbf{\Phi})$ and $L(\mathbf{\Phi})$, respectively for conciseness. The objective function $J(\mathbf{\Phi})$ contains the loss function $L(\mathbf{\Phi})$ computed over the training data $(\mathbf{x}, \mathbf{y})$ and the regularization term $N(\mathbf{\Phi})$ – this is known as the regularized *empirical risk* of the classifier $\mathbf{\Phi}$. In the following theorem, we establish a bound on the regularized empirical excess risk of the differentially private classifier minimizing the perturbed objective function $J_p(\mathbf{\Phi})$ over the classifier minimizing the unperturbed objective function $J(\mathbf{\Phi})$. We use the strong convexity of the objective function $J(\mathbf{\Phi})$ as given by Lemma A.2.2.

Theorem 4.1.3. *With probability at least $1 - \delta$, the regularized empirical excess risk of the classifier $\mathbf{\Phi}^p$ minimizing the perturbed objective function $J_p(\mathbf{\Phi})$ over the classifier $\mathbf{\Phi}^*$ minimizing the unperturbed objective function $J(\mathbf{\Phi})$ is bounded as*

$$J(\mathbf{\Phi}^p) \leq J(\mathbf{\Phi}^*) + \frac{8(d+1)^4 C}{\epsilon^2 \lambda} \log^2\left(\frac{d}{\delta}\right).$$

The upper bound on the regularized empirical risk is in $O(\frac{C}{\epsilon^2})$. The bound increases for smaller values of ϵ which implies tighter privacy and therefore suggests a trade off between privacy and utility.

The regularized empirical risk of a classifier is calculated over a given training dataset. In practice, we are more interested in how the classifier will perform on new test data which is assumed to be generated from the same source as the training data. The expected value of the loss function computed over the data is called the *true risk* $\tilde{L}(\mathbf{\Phi}) = \mathbb{E}[L(\mathbf{\Phi})]$ of the classifier $\mathbf{\Phi}$. In the following theorem, we establish a bound on the true excess risk of the differentially private classifier minimizing the perturbed objective function and the classifier minimizing the original objective function.

Theorem 4.1.4. *With probability at least $1-\delta$, the true excess risk of the classifier $\mathbf{\Phi}^p$ minimizing the perturbed objective function $J_p(\mathbf{\Phi})$ over the classifier $\mathbf{\Phi}^*$ minimizing the unperturbed objective function $J(\mathbf{\Phi})$ is bounded as*

$$\tilde{L}(\mathbf{\Phi}^p) \leq \tilde{L}(\mathbf{\Phi}^*) + \frac{4\sqrt{d}(d+1)^2 C}{\epsilon\lambda} \log\left(\frac{d}{\delta}\right) + \frac{8(d+1)^4 C}{\epsilon^2\lambda} \log^2\left(\frac{d}{\delta}\right) + \frac{16}{\lambda n}\left[32 + \log\left(\frac{1}{\delta}\right)\right].$$

Similar to the bound on the regularized empirical excess risk, the bound on the true excess risk is also inversely proportional to ϵ reflecting the privacy-utility trade-off. The bound is linear in the number of classes C, which is a consequence of the multi-class classification. The classifier learned using a higher value of the regularization parameter λ will have a higher covariance for each class ellipsoid. This would also make the classifier less sensitive to the perturbation. This intuition is confirmed by the fact that the true excess risk bound is inversely proportional to λ.

Experiments

We performed experiments with our differentially private large margin Gaussian classifier to observe how much error is introduced by the perturbation. We implemented the classifier using the `CVX` convex program solver [Grant and Boyd, 2008, 2010] .

We trained the classifier with the different random samples of the perturbation term b, each sampled with the increasing values of ϵ, and the regularization parameter λ = 0.01. For training data we used 500 data points with 10 dimensions equally split into the five classes each randomly sampled from the Gaussian distributions with unit variances. We tested the model on another 500 data points with 10 dimensions equally split into the five classes each also sampled from Gaussian distribution. The test error results averaged over 100 runs are shown in Figure 4.4.

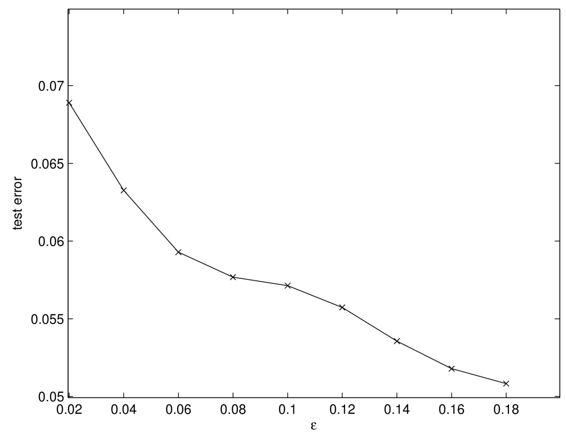

Figure 4.4: Test error vs. ϵ

We observe that for small value of ϵ implying tighter privacy constraints, we observe a higher error. By increasing ϵ, we see that the error steadily goes down as predicted by the bound on the true excess risk.

4.1.3 Differentially Private Classification from Multiparty Data

In [Pathak et al., 2010a], we address the problem of learning a differentially private classifier from a set of parties each possess data. The aim is to learn a classifier from the union of all the data. We specifically consider a logistic regression classifier, but as we shall see, the techniques are generally applicable to any classification algorithm. The conditions we impose are that (a) None of the parties are willing to share the data with one another or with any third party (*e.g.* a curator). (b) The computed classifier cannot be reverse engineered to learn about any individual data instance possessed by any contributing party.

We provide an alternative solution: within our approach the individual parties locally compute an optimal classifier with their data. The individual classifiers are then averaged to obtain the final aggregate classifier. The aggregation is performed through a secure protocol that also adds a stochastic component to the averaged classifier, such that the resulting aggregate classifier is differentially private, *i.e.*, no inference may be made about individual data instances from the classifier. This procedure satisfies both criteria (a) and (b) mentioned above. Furthermore, it is significantly less expensive than any SMC protocol to compute the classifier on the combined data.

Multiparty Classification Protocol

The problem we address is as follows: a number of parties $P_1, \ldots, P_K$ possess data sets $D_1, \ldots, D_K$ where $D_i = (\mathbf{x}, \mathbf{y})|_j$ includes a set of instances **x** and their binary labels **y**. We want to train a logistic regression classifier on the combined data such that no party is required to expose any of its data, and the no information about any single data instance can be obtained from the learned classifier. The protocol can be divided into the three following phases:

- **Training Local Classifiers on Individual Datasets.**

  Each party P_j uses their data set $(\mathbf{x}, \mathbf{y})|_j$ to learn an ℓ_2 regularized logistic regression classifier with weights $\hat{\mathbf{w}}_j$. This is obtained by minimizing the following objective function

$$\hat{\mathbf{w}}_j = \underset{\mathbf{w}}{\operatorname{argmin}}\ J(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}}\ \frac{1}{n_j} \sum_i \log\left(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}\right) + \lambda \mathbf{w}^T \mathbf{w}, \tag{4.14}$$

  where $\lambda > 0$ is the regularization parameter. Note that no data or information has been shared yet.

- **Publishing a Differentially Private Aggregate Classifier.** The proposed solution, illustrated by Figure 4.5, proceeds as follows. The parties then collaborate to compute an *aggregate* classifier given by $\hat{\mathbf{w}}^s = \frac{1}{K}\sum_j \hat{\mathbf{w}}_j + \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is a d-dimensional random variable sampled from a Laplace distribution scaled with the parameter $\frac{2}{n_{(1)}\epsilon\lambda}$ and $n_{(1)} = \min_j n_j$. As we shall see later, composing an aggregate classifier in this manner incurs only a well-bounded excess risk over training a classifier directly on the union of all data while enabling the parties to maintain their privacy. We also show that the noise term $\boldsymbol{\eta}$ ensures that the classifier $\hat{\mathbf{w}}^s$ satisfies differential privacy, *i.e.*, that individual data instances cannot be discerned from the aggregate classifier. The definition of the noise term $\boldsymbol{\eta}$ above may appear unusual at this stage, but it has an intuitive explanation: A classifier constructed by aggregating locally trained classifiers is limited by the performance of the individual classifier that has the least number of data instances. We note that the parties P_j cannot simply take their individually trained classifiers $\hat{\mathbf{w}}_j$, perturb them with a noise vector and publish the perturbed classifiers, because aggregating such classifiers will not give the correct $\boldsymbol{\eta} \sim \text{Lap}\big(2/(n_{(1)}\epsilon\lambda)\big)$ in general. Since individual parties cannot simply add noise to their classifiers to impose differential privacy, the actual averaging operation must be performed such that the individual parties do not expose their own classifiers or the number of data instances they possess. We therefore use a private multiparty protocol, interacting with an untrusted curator "Charlie" to perform the averaging. The outcome of the protocol is such that each of the parties obtain additive shares of the final classifier $\hat{\mathbf{w}}^s$, such that these shares must be added to obtain $\hat{\mathbf{w}}^s$.

  Privacy Preserving Protocol. We use asymmetric key additively homomorphic encryption [Paillier, 1999]. A desirable property of such schemes is that we can perform operations on the ciphertext elements which map into known operations on the same plaintext elements. For an additively homomorphic encryption function $\xi(\cdot)$,

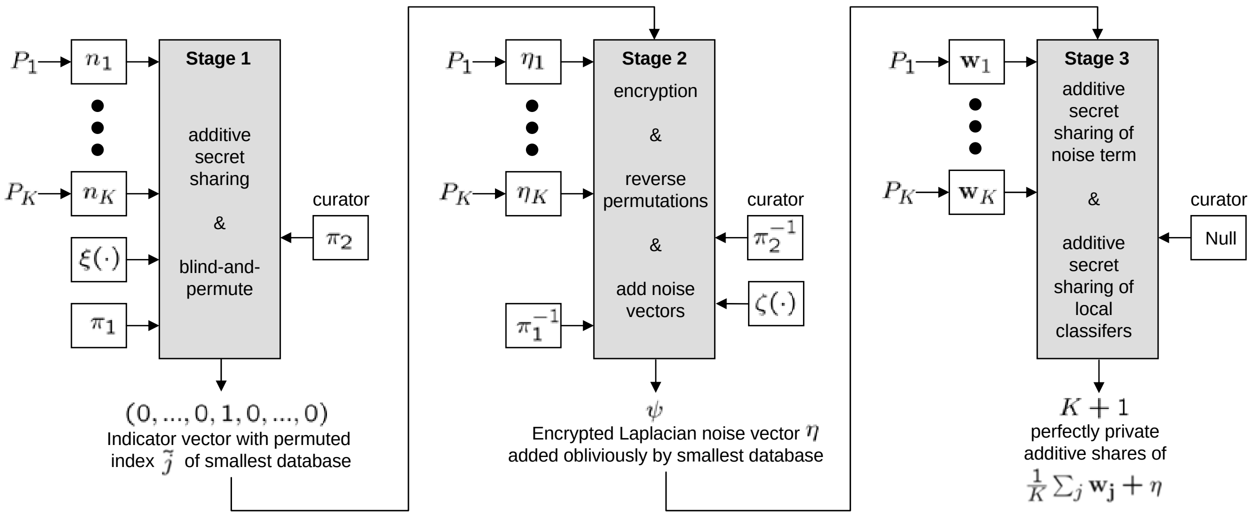
P_1
n_1
P_K
n_K
$\xi(\cdot)$
π_1
Stage 1
additive secret sharing & blind-and-permute
curator
π_2
$(0,...,0,1,0,...,0)$
Indicator vector with permuted index $\tilde{j}$ of smallest database
P_1
η_1
P_K
η_K
π_1^{-1}
Stage 2
encryption & reverse permutations & add noise vectors
curator
π_2^{-1}
$\zeta(\cdot)$
ψ
Encrypted Laplacian noise vector η added obliviously by smallest database
P_1
$\mathbf{w}_1$
P_K
$\mathbf{w}_K$
Stage 3
additive secret sharing of noise term & additive secret sharing of local classifers
curator
Null
$K+1$
perfectly private additive shares of
$\frac{1}{K}\sum_j \mathbf{w_j} + \eta$

Figure 4.5: Multiparty protocol to securely compute additive shares of $\hat{\mathbf{w}}^s$.

$\xi(a)\ \xi(b) = \xi(a+b),\ \ \xi(a)^b = \xi(ab)$. The additively homomorphic scheme employed here is semantically secure, i.e., repeated encryption of the same plaintext will result in different ciphertexts. For the ensuing protocol, encryption keys are considered public and decryption keys are privately owned by the specified parties. Assuming the parties to be honest-but-curious, the steps of the protocol are as follows.

Stage 1. Finding the index of the smallest database obfuscated by permutation.

1. Each party P_j computes $n_j = a_j + b_j$, where a_j and b_j are integers representing additive shares of the database lengths n_j for $j = 1, 2, \ldots, K$. Denote the K-length vectors of additive shares as **a** and **b** respectively.
2. The parties P_j mutually agree on a permutation π_1 on the index vector $(1, 2, \ldots, K)$. This permutation is unknown to Charlie. Then, each party P_j sends its share a_j to party $P_{\pi_1(j)}$, and sends its share b_j to Charlie with the index changed according to the permutation. Thus, after this step, the parties have permuted additive shares given by $\pi_1(\mathbf{a})$ while Charlie has permuted additive shares $\pi_1(\mathbf{b})$.
3. The parties P_j generate a key pair (pk,sk) where pk is a public key for homomorphic encryption and sk is the secret decryption key known only to the parties but not to Charlie. Denote element-wise encryption of **a** by $\xi(\mathbf{a})$. The parties send $\xi(\pi_1(\mathbf{a})) = \pi_1(\xi(\mathbf{a}))$ to Charlie.
4. Charlie generates a random vector $\mathbf{r} = (r_1, r_2, \cdots, r_K)$ where the elements r_i are integers chosen uniformly at random and are equally likely to be positive or negative. Then, he computes $\xi(\pi_1(a_j))\xi(r_j)$ = $\xi(\pi_1(a_j) + r_j)$. In vector notation, he computes $\xi(\pi_1(\mathbf{a}) + \mathbf{r})$. Similarly, by subtracting the same random integers in the same order to his own shares, he obtains $\pi_1(\mathbf{b}) - \mathbf{r}$ where π_1 was the permutation unknown to him and applied by the parties. Then, Charlie selects a permutation π_2 at random and obtains $\pi_2(\xi(\pi_1(\mathbf{a}) + \mathbf{r})) = \xi(\pi_2(\pi_1(\mathbf{a}) + \mathbf{r}))$ and $\pi_2(\pi_1(\mathbf{b}) - \mathbf{r})$. He sends $\xi(\pi_2(\pi_1(\mathbf{a}) + \mathbf{r}))$ to the individual parties in the following order: First element to P_1, second element to P_2,...,K^{th} element to P_K.
5. Each party decrypts the signal received from Charlie. At this point, the parties $P_1, P_2, \ldots, P_K$ respectively possess the elements of the vector $\pi_2(\pi_1(\mathbf{a}) + \mathbf{r})$ while Charlie possesses the vector $\pi_2(\pi_1(\mathbf{b}) - \mathbf{r})$. Since π_1 is unknown to Charlie and π_2 is unknown to the parties, the indices in both vectors have been complete obfuscated. Note also that, adding the vector collectively owned by the parties and the vector owned by Charlie would give $\pi_2(\pi_1(\mathbf{a})+\mathbf{r}) + \pi_2(\pi_1(\mathbf{b})-\mathbf{r}) = \pi_2(\pi_1(\mathbf{a}+\mathbf{b})) = \pi_2(\pi_1(\mathbf{n}))$. This situation in this step is similar to that encountered in the "blind and permute" protocol used for minimum-finding by Atallah and Li [2005].

6. Let $\pi_2(\pi_1(\mathbf{a})+\mathbf{r}) = \tilde{\mathbf{a}}$ and $\pi_2(\pi_1(\mathbf{b})-\mathbf{r}) = \tilde{\mathbf{b}}$. Then $n_i > n_j \Rightarrow \tilde{a}_i+\tilde{b}_i > \tilde{a}_j+\tilde{b}_j \Rightarrow \tilde{a}_i-\tilde{a}_j > \tilde{b}_j-\tilde{b}_i$. For each (i,j) pair with $i,j \in \{1,2,\ldots,K\}$, these comparisons can be solved by any implementation of a secure millionaire protocol [Yao, 1982]. When all the comparisons are done, Charlie finds the index $\tilde{j}$ such that $\tilde{a}_{\tilde{j}} + \tilde{b}_{\tilde{j}} = \min_j n_j$. The *true* index corresponding to the smallest database has already been obfuscated by the steps of the protocol. Charlie holds only an additive share of $\min_j n_j$ and thus cannot know the true length of the smallest database.

Stage 2. Obliviously obtaining encrypted noise vector from the smallest database.

1. Charlie constructs an K indicator vector $\mathbf{u}$ such that $u_{\tilde{j}} = 1$ and all other elements are 0. He then obtains the permuted vector $\pi_2^{-1}(\mathbf{u})$ where π_2^{-1} inverts π_2. He generates a key-pair (pk',sk') for additive homomorphic function $\zeta(\cdot)$ where only the encryption key pk' is publicly available to the parties P_j. Charlie then transmits $\zeta(\pi_2^{-1}(\mathbf{u})) = \pi_2^{-1}(\zeta(\mathbf{u}))$ to the parties P_j.
2. The parties mutually obtain a permuted vector $\pi_1^{-1}(\pi_2^{-1}(\zeta(\mathbf{u}))) = \zeta(\mathbf{v})$ where π_1^{-1} inverts the permutation π_1 originally applied by the parties P_j in Stage I. Now that both permutations have been removed, the index of the non-zero element in the indicator vector $\mathbf{v}$ corresponds to the true index of the smallest database. However, since the parties P_j cannot decrypt $\zeta(\cdot)$, they cannot find out this index.
3. For $j = 1,\ldots,K$, party P_j generates $\boldsymbol{\eta}_j$, a d-dimensional noise vector sampled from a Laplace distribution with parameter $\frac{2}{n_j\epsilon\lambda}$. Then, it obtains a d-dimensional vector $\boldsymbol{\psi}_j$ where for $i = 1,\ldots,d$, $\psi_j(i) = \zeta(v(j))^{\eta_j(i)} = \zeta(v(j)\ \eta_j(i))$.
4. All parties P_j now compute a d-dimensional noise vector $\boldsymbol{\psi}$ such that, for $i = 1,\ldots,d$, $\psi(i) = \prod_j \psi_j(i) = \prod_j \zeta(v(j)\eta_j(i)) = \zeta\left(\sum_j v(j)\eta_j(i)\right)$.
The reader will notice that, by construction, the above equation selects only the Laplace noise terms for the smallest database, while rejecting the noise terms for all other databases. This is because $\mathbf{v}$ has an element with value 1 at the index corresponding to the smallest database and has zeroes everywhere else. Thus, the decryption of $\boldsymbol{\psi}$ is equal to $\boldsymbol{\eta}$ which was the desired perturbation term defined at the beginning of this section.

Stage 3. Generating secret additive shares of $\hat{\mathbf{w}}^s$.

1. One of the parties, say P_1, generates a d-dimensional random integer noise vector $\mathbf{s}$, and transmits $\psi(i)\zeta(s(i))$ for all $i = 1,\ldots,d$ to Charlie. Using $\mathbf{s}$ effectively prevents Charlie from discovering $\boldsymbol{\eta}$, and therefore still ensures that no information is leaked about the database owners P_j. P_1 computes $\mathbf{w}_1 - K\mathbf{s}$.
2. Charlie decrypts $\psi(i)\zeta(s(i))$ to obtain $\eta(i) + s(i)$ for $i = 1,\ldots,d$. At this stage, the parties and Charlie have the following d-dimensional vectors: Charlie has $K(\boldsymbol{\eta} + \mathbf{s})$, P_1 has $\hat{\mathbf{w}}_1 - K\mathbf{s}$, and all other parties $P_j,\ j = 2,\ldots,K$ have $\hat{\mathbf{w}}_j$. None of the $K+1$ participants can share this data for fear of compromising differential privacy.
3. Finally, Charlie and the K database-owning parties run a simple secure function evaluation protocol [Ben-Or et al., 1988], at the end of which each of the $K+1$ participants obtains an additive share of $K\hat{\mathbf{w}}^s$. This protocol is provably private against honest but curious participants when there are no collisions. The resulting shares are published.

The above protocol ensures the following (a) None of the $K+1$ participants, or users of the perturbed aggregate classifier can find out the size of any database, and therefore none of the parties knows who contributed $\boldsymbol{\eta}$ (b) Neither Charlie nor any of the parties P_j can individually remove the noise $\boldsymbol{\eta}$ after the additive shares are published. This last property is important because if anyone knowingly could remove the noise term, then the resulting classifier no longer provides differential privacy.

- **Testing Phase.** A test participant Dave having a test data instance $\mathbf{x}' \in \mathbb{R}^d$ is interested in applying the trained classifier adds the published shares and divides by K to get the differentially private classifier $\hat{\mathbf{w}}^s$. He can then compute the sigmoid function $t = \frac{1}{1+e^{-\hat{\mathbf{w}}^{sT}\mathbf{x}_i}}$ and decide to classify $\mathbf{x}'$ with label -1 if $t \leq \frac{1}{2}$ and with label 1 if $t > \frac{1}{2}$.

Proof of Differential Privacy

We show that the perturbed aggregate classifier satisfies differential privacy. We use the following bound on the sensitivity of the regularized regression classifier as proved in Corollary 2 in [Chaudhuri and Monteleoni, 2008] restated in the appendix as Theorem A.3.1.

Theorem 4.1.5. *The classifier $\hat{\boldsymbol{w}}^s$ preserves ϵ-differential privacy. For any two adjacent datasets D and D',*

$$\left| \log \frac{P\left(\hat{\boldsymbol{w}}^s | D\right)}{P\left(\hat{\boldsymbol{w}}^s | D'\right)} \right| \leq \epsilon.$$

Analysis of Excess Risk

In the following discussion, we consider how much excess error is introduced when using a perturbed aggregate classifier $\hat{\mathbf{w}}^s$ satisfying differential privacy as opposed to the unperturbed classifier $\mathbf{w}^*$ trained on the entire training data while ignoring the privacy constraints as well as the unperturbed aggregate classifier $\hat{\mathbf{w}}$.

We first establish a bound on the ℓ_2 norm of the difference between the aggregate classifier $\hat{\mathbf{w}}$ and the classifier $\mathbf{w}^*$ trained over the entire training data. To prove the bound we apply Lemma 1 from [Chaudhuri and Monteleoni, 2008] restated as Lemma A.3.2 in the appendix. Please refer to the appendix for the proof of the following theorem.

Theorem 4.1.6. *Given the aggregate classifier $\hat{\boldsymbol{w}}$, the classifier $\boldsymbol{w}^*$ trained over the entire training data and $n_{(1)}$ is the size of the smallest training dataset,*

$$\|\hat{\boldsymbol{w}} - \boldsymbol{w}^*\|_2 \leq \frac{K-1}{n_{(1)}\lambda}.$$

The bound is inversely proportional to the number of instances in the smallest dataset. This indicates that when the datasets are of disparate sizes, $\hat{\mathbf{w}}$ will be a lot different from $\mathbf{w}^*$. The largest possible value for $n_{(1)}$ is $\frac{n}{K}$ in which case all parties having an equal amount of training data and $\hat{\mathbf{w}}$ will be closest to $\mathbf{w}^*$. In the one party case for $K = 1$, the bound indicates that norm of the difference would be upper bounded by zero, which is a valid sanity check as the aggregate classifier $\hat{\mathbf{w}}$ is the same as $\mathbf{w}^*$.

We use this result to establish a bound on the empirical risk of the perturbed aggregate classifier $\hat{\mathbf{w}}^s = \hat{\mathbf{w}} + \boldsymbol{\eta}$ over the empirical risk of the unperturbed classifier $\mathbf{w}^*$ in the following theorem. Please refer to the appendix for the proof.

Theorem 4.1.7. *If all data instances $\boldsymbol{x}_i$ lie in a unit ball, with probability at least $1-\delta$, the empirical regularized excess risk of the perturbed aggregate classifier $\hat{\boldsymbol{w}}^s$ over the classifier $\boldsymbol{w}^*$ trained over entire training data is*

$$J(\hat{\boldsymbol{w}}^s) \leq J(\boldsymbol{w}^*) + \frac{(K-1)^2(\lambda+1)}{2n_{(1)}^2\lambda^2} + \frac{2d^2(\lambda+1)}{n_{(1)}^2\epsilon^2\lambda^2}\log^2\left(\frac{d}{\delta}\right) + \frac{2d(K-1)(\lambda+1)}{n_{(1)}^2\epsilon\lambda^2}\log\left(\frac{d}{\delta}\right).$$

The bound suggests an error because of two factors: aggregation and perturbation. The bound increases for smaller values of ϵ implying a tighter definition of differential privacy, indicating a clear trade-off between privacy and utility. The bound is also inversely proportional to $n_{(1)}^2$ implying an increase in excess risk when the parties have training datasets of disparate sizes.

In the limiting case $\epsilon \to \infty$, we are adding a perturbation term $\boldsymbol{\eta}$ sampled from a Laplacian distribution of infinitesimally small variance resulting in the perturbed classifier being almost as same as using the unperturbed aggregate classifier $\hat{\mathbf{w}}$ satisfying a very loose definition of differential privacy. With such a value of ϵ, our bound becomes

$$J(\hat{\mathbf{w}}) \leq J(\mathbf{w}^*) + \frac{(K-1)^2(\lambda+1)}{2n_{(1)}^2\lambda^2}. \tag{4.15}$$

Similar to the analysis of Theorem 4.1.6, the excess error in using an aggregate classifier is inversely proportional to the size of the smallest dataset $n_{(1)}$ and in the one party case $K = 1$, the bound becomes zero as the aggregate classifier $\hat{\mathbf{w}}$ is the same as $\mathbf{w}^*$. Also, for a small value of ϵ in the one party case $K = 1$ and $n_{(1)} = n$, our bound reduces to that in Lemma 3 of [Chaudhuri and Monteleoni, 2008],

$$J(\hat{\mathbf{w}}^s) \leq J(\mathbf{w}^*) + \frac{2d^2(\lambda+1)}{n^2\epsilon^2\lambda^2} \log^2\left(\frac{d}{\delta}\right). \tag{4.16}$$

While the previous theorem gives us a bound on the empirical excess risk over a given training dataset, it is important to consider a bound on the true excess risk of $\hat{\mathbf{w}}^s$ over $\mathbf{w}^*$. Let us denote the true risk of the classifier $\hat{\mathbf{w}}^s$ by $\tilde{J}(\hat{\mathbf{w}}^s) = \mathbb{E}[J(\hat{\mathbf{w}}^s)]$ and similarly, the true risk of the classifier $\mathbf{w}^*$ by $\tilde{J}(\mathbf{w}^*) = \mathbb{E}[J(\mathbf{w}^*)]$. In the following theorem, we apply the result from [Sridharan et al., 2008] which uses the bound on the empirical excess risk to form a bound on the true excess risk. Please refer to the appendix for the proof.

Theorem 4.1.8. *If all training data instances $\boldsymbol{x}_i$ lie in a unit ball, with probability at least $1 - \delta$, the true excess risk of the perturbed aggregate classifier $\hat{\boldsymbol{w}}^s$ over the classifier $\boldsymbol{w}^*$ trained over entire training data is*

$$\begin{aligned}\tilde{J}(\hat{\boldsymbol{w}}^s) \leq \tilde{J}(\boldsymbol{w}^*) &+ \frac{2(K-1)^2(\lambda+1)}{2n_{(1)}^2\lambda^2} + \frac{4d^2(\lambda+1)}{n_{(1)}^2\epsilon^2\lambda^2}\log^2\left(\frac{d}{\delta}\right) \\ &+ \frac{4d(K-1)(\lambda+1)}{n_{(1)}^2\epsilon\lambda^2}\log\left(\frac{d}{\delta}\right) + \frac{16}{\lambda n}\left[32 + \log\left(\frac{1}{\delta}\right)\right].\end{aligned}$$

Experiments

We perform an empirical evaluation of the proposed differentially private classifier to obtain a characterization of the increase in the error due to perturbation. We use the Adult dataset from the UCI machine learning repository [Frank and Asuncion, 2010] consisting of personal information records extracted from the census database and the task is to predict whether a given person has an annual income over \$50,000. The choice of the dataset is motivated as a realistic example for application of data privacy techniques. The original Adult data set has six continuous and eight categorical features. We use pre-processing similar to [Platt, 1999], the continuous features are discretized into quintiles, and each quintile is represented by a binary feature. Each categorical feature is converted to as many binary features as its cardinality. The dataset contains 32,561 training and 16,281 test instances each with 123 features.[1] In Figure 4.6, we compare the test error of perturbed aggregate classifiers trained over data from five parties for different values of ϵ. We consider three situations: all parties with equal datasets containing 6512 instances (even split, $n_{(1)}$ = 20% of n), parties with datasets containing 4884, 6512, 6512, 6512, 8141 instances ($n_{(1)}$ = 15% of n), and parties with datasets containing 3256, 6512, 6512, 6512, 9769 instances ($n_{(1)}$ = 10% of n). We also compare with the error of the classifier trained using combined training data and its perturbed version satisfying differential privacy. We chose the value of the regularization parameter $\lambda = 1$ and the results displayed are averaged over 200 executions.

The perturbed aggregate classifier which is trained using maximum $n_{(1)} = 6512$ does consistently better than for lower values of $n_{(1)}$ which is same as our theory suggested. Also, the test error for all perturbed aggregate classifiers drops with ϵ, but comparatively faster for even split and converges to the test error of the classifier trained over the combined data. As expected, the differentially private classifier trained over the entire training data does much better than the perturbed aggregate classifiers with an error equal to the unperturbed classifier except for small values of ϵ. The lower error of this classifier is at the cost of the loss in privacy of the parties as they would need to share the data in order to train the classifier over combined data.

4.2 Applications

4.2.1 Privacy Preserving Keyword Recognition

In [Pathak et al., 2010b], we develop a privacy-preserving framework for hidden Markov model inference. We consider the multiparty scenario in which the data and the HMMs belong to different individuals and cannot be shared. For

[1]The dataset can be download from http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/binary.html#a9a

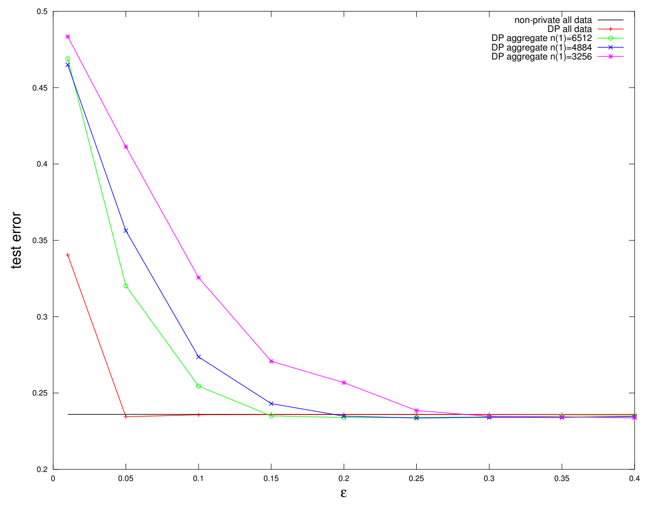

Figure 4.6: Classifier performance evaluated for $\mathbf{w}^*$, $\mathbf{w}^* + \boldsymbol{\eta}$, and $\hat{\mathbf{w}}^s$ for different data splits vs. ϵ

example, Alice (the client) wants to conduct analysis on speech data from telephone calls. She outsources the speech recognition task to Bob (the server), who possesses an accurate HMM obtained via extensive training. Alice cannot share the speech data with Bob owing to privacy concerns while Bob cannot disclose the model parameters because this might leak valuable information about his own training database.

Smaragdis and Shashanka [2007] present the only major work on privacy preserving HMMs using cryptographic primitives. Those protocols are designed such that both the parties perform equal amount of computation. In contrast, we present asymmetric protocols that are more suitable for client-cloud interactions; the client encrypts the data and provides it to a server in the cloud which shoulders most of the computational burden. Further, since HMM-based probabilistic inference manipulates probabilities and real numbers in general, the effect of finite precision, underflow and exponentiation in the ciphertext domain are considered here for the first time.

Protocol for Secure Forward Algorithm

In the following, assume that Alice (a client) sets up a private and public key pair for the Paillier encryption function $\xi(\cdot)$, and sends only the public key to Bob (a server). Consider the following protocols

Secure Logarithm Protocol

Input: Bob has input $\xi(\theta)$;
Output: Bob obtains the encryption $\xi(\log\theta)$ of $\log\theta$, and Alice obtains no information about θ.

1. Bob randomly chooses β, and sends $\xi(\theta)^\beta = \xi(\beta\theta)$ to Alice
2. Alice decrypts $\beta\theta$, and then sends $\xi(\log\beta\theta)$ to Bob.
3. Bob computes $\xi(\log\beta\theta)\cdot\xi(-\log\beta) = \xi(\log\theta + \log\beta)\cdot\xi(-\log\beta) = \xi(\log\theta)$

Secure Exponent Protocol

Input: Bob has input $\xi(\log\theta)$,
Output: Bob obtains the encryption $\xi(\theta)$, and Alice obtains no information about θ.

1. Bob randomly chooses $\beta \in \mathbb{Z}^*_{N^2}$ and sends $\xi(\log_g\theta)\xi(\log_g\beta) = \xi(\log_g\theta + \log_g\beta) = \xi(\log_g\beta\theta)$ to Alice.

2. Alice decrypts $\log_g \beta\theta$, and then sends $\xi(\beta\theta)$ to Bob

3. Bob computes $\xi(\beta\theta)^{\frac{1}{\beta}} = \xi(\theta)$ where $\frac{1}{\beta}$ is the multiplicative inverse of β in $\mathbb{Z}^*_{N^2}$.

Secure LOGSUM Protocol

Input: Bob has $(\xi(\log\theta_1), \xi(\log\theta_2), \ldots, \xi(\log\theta_n))$ and $(a_1, a_2, \ldots, a_n)$;
Output: Bob obtains $\xi(\log\sum_{i=1}^n a_i\theta_i)$, and Alice discovers nothing about the θ_i

1. With Bob's input $\xi(\log\theta_1), \xi(\log\theta_2), \ldots, \xi(\log\theta_n)$, Alice and Bob execute the Secure Exponent protocol repeatedly so that Bob obtains $\xi(\theta_1), \xi(\theta_2), \ldots, \xi(\theta_n)$.

2. Bob exploits the additive homomorphic property of Paillier encryption to obtain

$$\xi\left(\sum_{i=1}^n a_i\theta_i\right) = \prod_{i=1}^n \xi(a_i\theta_i) = \prod_{i=1}^n \xi(\theta_i)^{a_i}$$

3. Bob and Alice execute the Secure Logarithm protocol, at the end of which Bob obtains the encryption $\xi(\log\sum_{i=1}^n a_i\theta_i)$.

Secure Forward Algorithm Protocol

Input: Alice has an observation sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$. Bob has the HMM $\lambda = (A, B, \Pi)$.
Output: Bob obtains $\xi(\log \Pr\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T|\lambda)\}$.

Write the matrix B as $[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N]$, where for each $j = 1, 2, ..., N$, $\mathbf{b}_j$ is a column vector with component $b_j(v_k)$, $k = 1, 2, ..., M$. Now, a privacy-preserving version of the forward algorithm for HMMs proceeds as follows:

1. For each $t = 1, 2, ..., T$ and $j = 1, 2, ..., N$, Bob randomly chooses γ_{tj} and generates a column vector $\log\mathbf{b}_j + \gamma_{tj}$.

2. Based on the input $\mathbf{x}_t$, Alice uses 1-of-M OT to obtain $\log b_j(\mathbf{x}_t) + \gamma_{tj}$.

3. Alice sends $\xi(\log b_j(\mathbf{x}_t) + \gamma_{tj})$ to Bob

4. Using γ_{tj} and the homomorphic property, Bob computes $\xi(\log b_j(\mathbf{x}_t) + \gamma_{tj}) \cdot \xi(-\gamma_{tj}) = \xi(\log b_j(\mathbf{x}_t))$ for $j = 1, 2, ..., N, t = 1, 2, ..., T$.

5. Bob computes $\xi(\log\alpha_1(j)) = \xi(\log\pi_j) \cdot \xi(\log b_j(\mathbf{x}_1))$ for $j = 1, 2, ..., N$

6. Induction Step: For $j = 1, 2, ..., N$, with Bob's input $\xi(\log\alpha_t(j))$, $j = 1, 2, ..., N$ and the transition matrix $A = (a_{ij})$, Alice and Bob run the secure LOGSUM protocol, at the end of which Bob obtains $\xi(\log\sum_{l=1}^N \alpha_t(l)a_{lj})$.

7. For all $1 \leq t \leq T-1$, Bob computes

$$\xi(\log\alpha_{t+1}(j)) = \xi(\log\sum_{l=1}^N \alpha_t(l)a_{lj}) \cdot \xi(\log b_j(\mathbf{x}_{t+1}))$$

8. Using the additively homomorphic property, Bob determines $\xi(\log P(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T|\lambda)) = \prod_{j=1}^N \xi(\log\alpha_T(j))$

Security Analysis

The Secure Logarithm, Secure Exponent and Secure LOGSUM protocols rely on multiplicative secret sharing to prevent Alice from discovering θ. Bob cannot discover θ because he does not possess the decryption key for the Paillier cryptosystem. The security of the Forward Algorithm protocol derives from the security of the previous three primitive protocols and the security of 1-of-M OT.

Secure Keyword Recognition

Consider the following scenario for privacy-preserving keyword recognition: Alice has a sampled speech signal, which she converts into T frames, where each frame is represented by a 39-dimensional vector of Mel Frequency Cepstral Coefficients (MFCCs). Thus Alice possesses $\mathbf{x}_i$, $i = 1, 2, ..., T$ where each $\mathbf{x}_i \in \mathbb{R}^{39}$. Derivation of MFCCs from speech signals is an established practice in the speech processing literature Rabiner and Juang [1993]. Bob possesses Δ different HMMs, each trained for a single keyword. Alice and Bob will execute a secure protocol, at the end of which, Alice will discover the keyword that is most likely to be contained in her speech sample.

Let $d = 39$. Let a d-dimensional vector $\mathbf{y}$ of MFCCs have a multivariate Gaussian distribution, i.e., $b_j(\mathbf{y}) = \mathcal{N}(\mu_j, \mathbf{C}_j)$, where $j = 1, 2, ..., N$ indexes the state of a HMM λ. Let $\mathbf{z} = [\mathbf{y}^T, 1]^T$. Then, $\log b_j(\mathbf{y}) = \mathbf{z}^T \mathbf{W}_j \mathbf{z}$, where

$$\mathbf{W}_j = \left[\begin{array}{c|c} -\frac{1}{2}\mathbf{C}_j^{-1} & \mathbf{C}_j^{-1}\mu_j \\ \hline 0 & w_j \end{array}\right] \in \mathbb{R}^{(d+1)\times(d+1)}$$

and $w_j = \frac{1}{2}\mu_j^T \mathbf{C}_j^{-1}\mu_j - \frac{1}{2}\ln|\mathbf{C}_j^{-1}| - \frac{d}{2}\ln 2\pi$. The above treatment considers $\mathbf{y}$ to be a single multivariate Gaussian random variable, though an extension to mixture of multivariate Gaussians is also possible. Note that the matrix $\mathbf{W}_j$ is available to Bob. Further, note that $\log b_j(\mathbf{y})$ is a linear function of products $z_i z_j$ where $i, j \in \{1, 2, ..., d+1\}$. This allows us to simplify the Secure Forward Algorithm Protocol as follows:

Simplified Secure Forward Algorithm

1. For each $t = 1, 2, ..., T$, Alice sets $\mathbf{z} = [\mathbf{x}_t^T, 1]^T$. Alice sends to Bob the encryptions of all $z_i z_j$ with $i, j \in \{1, 2, ..., d+1\}$.
2. For each HMM state $j = 1, 2, ..., N$, Bob obtains the encryption $\xi(\log b_j(\mathbf{x}_t)) = \xi(\mathbf{z}^T \mathbf{W}_j \mathbf{z})$ using the additive homomorphic property.
3. Bob executes steps 5–8 of the secure forward algorithm protocol.

Now, the protocol for secure keyword recognition is as follows.

Protocol for Secure Keyword Recognition

Input: Alice has the MFCC feature vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$ corresponding to her privately owned speech sample. Bob has the database of HMMs $\{\lambda_1, \lambda_2, ..., \lambda_\Delta\}$ where each HMM λ_δ corresponds to a single keyword, which is denoted by τ_δ;
Output: Alice obtains $\delta^* = \arg\max_\delta \Pr\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_T}|\lambda_\delta\}$.

1. Alice and Bob execute the simplified secure forward algorithm protocol for each HMM λ_δ, at the end of which Bob obtains $\xi(p_\delta) = \xi(\log \Pr\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x_T}|\lambda_\delta\})$, $\delta = 1, 2, ..., \Delta$.
2. Bob chooses an order-preserving matrix[2] $R = (r_{ij})_{\Delta\times\Delta}$ with random coefficients. Using the additively homomorphic property of Paillier encryption, he computes the element-wise encryption given by $(\xi(p'_1), ..., \xi(p'_\Delta)) = (\xi(p_1), ..., \xi(p_\Delta)) \cdot R$. Bob sends the result to Alice.
3. Alice decrypts and obtains $\delta^* = \max_\delta p'_\delta = \max_\delta p_\delta$. This is true because R is order-preserving mapping.
4. Alice and Bob perform a 1-of-Δ OT, and Alice obtains the word τ_{δ^*}.

[2]See Rane and Sun [2010] for more details

Computational Complexity

Let us consider the computational overhead of the protocol in terms of the number of keywords (Δ), the number of HMM states (N), the time duration of the observation (T) and the size of each observation (d). Alice has an overhead of $O(d^2T)$ encryptions at the beginning of the protocol. This initial computational overhead is independent of the number of keywords and the size of the HMM. Bob has an overhead of $O(d^2N\Delta T)$ ciphertext multiplications and exponentiations in order to determine the $\xi(\log b_j(\mathbf{x}_t))$ terms. Finally, to determine the $\xi(\alpha_T(j))$ terms, Alice and Bob both have an overhead of $O(N\Delta T)$. Thus Alice's protocol overhead is $O((d^2 + N\Delta)T)$ while Bob's overhead is $O(d^2N\Delta T)$, which means that Bob's overhead grows faster than Alice's.

Practical Issues: Fixed Precision and Numerical Underflow

The Paillier cryptosystem is defined over an integer ring but the speech data consists of real numbers causing the protocol to introduce an error in the computation. In our implementation, we used numbers accurate to 6 decimal places, which means that all floating point values were multiplied by a constant value 10^6 before encryption and divided by the same value after decryption. Furthermore, whenever there was a floating point number in the ciphertext exponent – of the form $\xi(a)^b$ – then, the fractional part was truncated, and only the integer portion of b was used in the calculation. These approximations introduce errors in the protocol which propagate during the iterations of the forward algorithm.

When implementing the protocol in software, another problem is that the value of $\alpha_t(j)$ gets progressively smaller after each iteration, and causes an underflow after a only a few iterations. This problem would not be encountered if the probabilities were always expressed in the form $\log \alpha_t(j)$. However, it is necessary to consider the actual value of $\alpha_t(j)$; the underflow problem is resolved by normalizing $\alpha_t(1), \ldots, \alpha_t(N)$ in each iteration, such that the relative values are preserved.

Experimental Analysis

We performed speech recognition experiments on a 3.2 GHz Intel Pentium 4 machine with 2 GB RAM and running 64-bit GNU/Linux. We created an efficient C++ implementation of the Paillier cryptosystem using the variable precision integer arithmetic library provided by OpenSSL. The encryption and decryption functions were used in a software implementation of the protocol.

The experiment was conducted on audio samples of length 0.98 seconds each. The samples consisted of spoken words which were analyzed by ten HMMs each having 5 states. These 10 HMMs were assumed to be trained offline (by Bob) on ten different words, viz., utterances of "one", "two", "three", and so on up to "ten". To perform speech recognition, each test speech sample was converted into 98 overlapping vectors (called frames), a 39-dimensional feature vector composed of MFCCs were extracted from each overlapping vector, as is common in speech processing. These 39-dimensional vectors serve as the input observations $\mathbf{x}_i$ owned by Alice.

The time required for processing an audio sample of length 0.98 seconds using the 10 HMMs was measured. The results are tabulated in Table 4.1 for the case of using 256-bit and 512-bit keys for Paillier homomorphic encryption. Using a longer key provides a stronger degree of security, but this it comes at the cost of a significant increase in processing time. Note that, once Bob receives the encrypted input from Alice, the calculation of $\xi(\log b_j(\mathbf{x}_t))$ for all 10 HMMs can proceed in parallel. This is possible, for instance, in a cloud computing framework, where Bob has access to several server nodes.

Activity	**256-bit keys**	**512-bit keys**
Alice encrypts input data (only once)	205.23 s	1944.27 s
Bob computes $\xi(\log b_j(\mathbf{x}_t))$ (per HMM)	79.47 s	230.30 s
Both compute $\xi(\alpha_T(j))$ (per HMM)	16.28 s	107.35 s

Table 4.1: Protocol execution times in seconds. Note that Alice's encryption operation occurs only once at the beginning of the protocol. The larger the dictionary of keywords, the larger is the computational overhead for Bob.

In order to observe the effect of error propagation, we compared the values of $\alpha_T(N)$ for a given 0.98-second speech sample derived using the the privacy preserving speech recognition protocol as well as a conventional non-secure implementation which uses no encryption. This comparison of $\alpha_T(N)$ with and without encryption was performed for each of the 10 HMMs corresponding to the spoken words“one” through “ten”. The relative error in $\alpha_T(N)$ with respect to the true values was found to be 0.5179% for both 256-bit and 512-bit Paillier encryption.

4.2.2 A framework for implementing SMC protocols

We created a C++ framework for quickly implementing SMC protocols for machine learning and speech processing algorithms. As we had seen before, the biggest bottleneck in implementing SMC protocols is the computational cost of the encryption and decryption functions and similarly, we require efficient numerical computation and linear algebra for implementing machine learning algorithms.

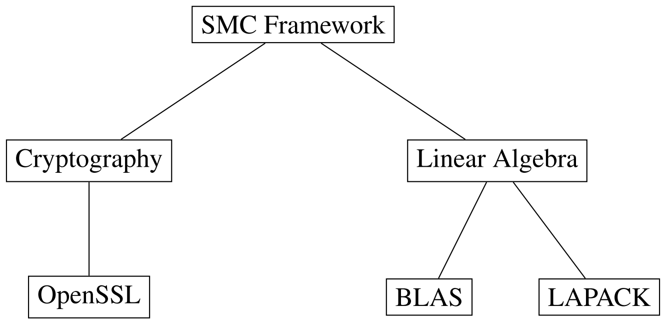

Figure 4.7: Architecture of the SMC framework

The main computational cost of the cryptosystem comes from the variable precision arithmetic operations. We used the variable precision arithmetic library provided by OpenSSL to implement the Paillier cryptosystem including homomorphic operations on the encrypted data such as modular multiplication and exponentiations of ciphertexts. Table 4.2 summarizes the time required to encrypt one number using our Paillier implementation across varying key sizes.

key size	**keygen**	**1000 encryptions**	**# encryption/s**
64 bits	0.006 s	0.063 s	15873.02
128 bits	0.022 s	0.213 s	4694.84
256 bits	0.174 s	1.207 s	828.50
512 bits	0.644 s	8.339 s	119.92
1024 bits	42.757 s	62.907 s	15.90

Table 4.2: Encryption speeds.

We used BLAS and LAPACK libraries for fast matrix operations such as matrix inversion, multiplication, determinant computation. we chose to use C++ because using a native implementation of these primitives is significantly faster than using an virtual machine based implementation like Java or an interpreted one like Python. We chose these libraries because they are ubiquitously available on most platforms and our framework can be used with just a compilation.

Our objective was to create an efficient framework with an high level interface which abstracts most of these details from the end developer. From our experiences in implementing SMC protocols using this framework, we believe that it would greatly benefit the research community.

Chapter 5

Proposed Work

In this section, we describe some of the ongoing work that we propose to investigate in the scope of this thesis. Some of the work is labeled optional in the sense that we recognize them as important future directions in our research and we hope to make progress as much as possible in the available time.

5.1 Techniques

5.1.1 Differentially Private Hidden Markov Models

We plan to investigate differentially private hidden Markov models (HMMs). Using this framework, it will be possible for a party to publish an HMM trained over private audio data. Such a release mechanism can be used as a guarantee of privacy while collecting audio data from participants for training speech recognition systems. In general, the differentially private formulation will greatly simplify the SMC protocols for speech recognition. The party with the differentially private HMM can simply transfer the model parameters to the party with the data who can perform the HMM inference necessary for speech recognition locally without the need for any data transmission.

A continuous density HMM is constituted by a finite set of states with an emission model giving the probability of a state emitting an observation and a transition model giving the probability of transitioning from one state to another. Our previous work on differentially private large margin GMM [Pathak and Raj, 2010b] can directly be used to model the emission probabilities. We can apply it in an iterative algorithm to learn the transition probabilities. Also, we plan to investigate differentially private mechanisms for the large margin HMM [Sha and Saul, 2007] and other structured classification algorithms such as max-margin Markov networks [Taskar et al., 2003].

5.1.2 Differential Private Classification by Transforming the Loss Function

We have seen two mechanisms to create differentially private classifiers – the exponential mechanism [Dwork et al., 2006] and the sensitivity method [Chaudhuri and Monteleoni, 2008]. The exponential mechanism proposed by adds perturbation to the resultant optimal solution so that the classifiers trained on adjacent datasets are likely to have the same optimum. The sensitivity method perturbs the convex objective function by a linear term to achieve the same result. We propose to investigate the following mechanisms.

1. **Adding randomization to the data.** This technique involves modifying the loss function by adding random perturbation to the data such that the resultant classifier satisfies differential privacy. One advantage of this approach is that it will naturally allow multiple parties to contribute the data.

2. **Kernel methods.** Here we modify the loss function by adding perturbation in the kernel itself. As the dual formulation of classifiers such as support vector machines involves the kernel trick dealing with only the dot products, adding noise perturbing the kernel inner product should be adequate to satisfy differential privacy and would introduce a much lower excess error.

In our work on differentially private classification, the ultimate goal would be to create a mechanism which preserves differential privacy while introducing the least amount of excess error and also to investigate under what situations would such a universally optimum mechanism exist. We propose to investigate this question in the context of large margin classification. Our intuition is that compared to other classification algorithms, a large margin classifier should be much more robust to perturbation.

5.1.3 Optional: Privately Training Classifiers over Data from Different Distributions

In [Pathak et al., 2010a], we considered the problem of training a differentially private classifier by aggregating locally trained classifiers from multiple parties. In this setting, our main assumption was that the data instances though belonging to multiple parties were sampled from the same source distribution. This assumption is found to be too restrictive in practice especially for problem areas such as speech processing as the individual characteristics of the participants play an important role. We aim to develop a differentially private aggregation mechanism where data belonging to each party is assumed to be sampled from different distribution.

5.2 Applications

5.2.1 Privacy Preserving Music Recognition and Keyword Spotting

In the keyword spotting involves detecting the presence of a keyword in an audio sample. A keyword is typically modeled as an HMM which a party can privately train using a audio dataset belonging to another party using a training protocol for speech recognition as described above. Using this model, we aim to construct an SMC protocol for iterative Viterbi decoding that efficiently finds the sub-sequence an observation sequence having the highest average probability of being generated by a given HMM. Additionally, we can also apply some of these techniques towards privately searching text documents, towards creating a "private grep."

Keyword spotting systems that use word level HMMs are analogous to HMM-based speaker identification systems in their operation, and would require similar approaches to them privately. However, when the system permits the user to specify the words and composes models for them from sub-word units an additional complexity is introduced if it is required that the identity of the keyword be hidden from the system. The approach will require composition of uninformative sub-word graphs and performing the computation in a manner that allows the recipient of the result to obtain scores from only the portions of the graph that represent the desired keyword. Ezzat and Poggio [2008] propose a more fruitful approach where the keyword spotting problem is reduced to computation of correlations and a simple SVM or boosted classifier. The approach is reportedly more accurate than the HMM-based system. Additionally, due to its simplicity, the complexity of privacy-preserving computation is also much lower for the method.

5.2.2 Privacy Preserving Speaker Verification and Identification

We are interested in constructing SMC protocols for speaker identification problem, where one party – the client has access to the audio data and another party – the server, wants to learn a multi-class classifier, typically, a Gaussian mixture model, to label each audio sample according to its speaker. Here are some of the issues we propose to consider.

1. **SMC Protocols for Speaker Verification.** In this case, the simplest systems that can be converted to privacy-preserving form are based on GMMs and HMMs. Speaker verification schemes employ a variety of "anti-speaker" and "garbage" models. Scores computed from the various models contribute to normalization schemes that must also be incorporated securely as done by [Bimbot et al., 2004]. These normalization schemes sometimes include selection of the models themselves that will used by the system. Also, in the more complex scenario, it may be desired that the user is correctly verified, but the system itself remains unaware of who exactly was verified. To do so, the privacy-preserving framework must keep all model types active at all times and the precise model used must not be revealed at any time.

2. **SMC Protocols for Speaker Identification** Given the similarity to speaker verification, the approach taken to the development of privacy preserving speaker identification systems will also be similar. However, in this case

we have an additional level of complexity of requiring that the number of speakers must also be hidden. Also, the actual outcome of the identification may need to be handled differently – the intended recipient might be a third party.

3. **Releasing Differentially Private Speaker Identification/Verification Classifier.** The speaker identification or verification problem involves training a classifier, which is typically a Gaussian mixture model, for validating a speaker's identity using an audio sample. If there is such a classifier trained over private collection of speech recordings, it can result in disclosure of information about the data. This limits the amount of analysis that can be done on such data as well as the prospects of their auxiliary information being revealed inhibits participants from contributing their audio data to such collections. Towards this, we created a mechanism for releasing a differentially private speaker identification classifier.

5.2.3 Optional: Privacy Preserving Graph Search for Speech Recognition

We are interested in constructing a cryptographic protocol for graph search problems. Most problems in speech recognition including model evaluation algorithms like dynamic time warping (DTW), Viterbi decoding, as well as model training algorithms such as Baum-Welch can be formulated as graph search problems; an efficient protocol for privacy preserving graph search can result in practical solutions for many of these problems.

The two specific directions we plan to investigate are:

1. **Graph Traversal**. We plan to study the general problem where one party has a graph and another party is interested in performing a graph traversal privately. While this is applicable in continuous speech recognition, we also plan to apply it to the general problem of matching finite automata and regular expressions with privacy constraints.

2. **Belief Propagation**. Given a graph structure, we are interested in evaluating how a potential function such as the probability of having a contagious disease spreads across the nodes in a private manner, *i.e.*, without any node knowing anything about the value of the function at its neighbors. Kearns et al. [2007] present a simplified privacy preserving message passing solution for undirected trees. We plan to extend it to general graphs.

5.3 Estimated Timeline

We plan to make progress towards the proposed work using the following timeline as a guide.

Time Slot	**Task**
Dec 2010 - Apr 2011	Privacy Preserving Music Matching & Keyword Spotting
May - Aug 2011	Differential Private Classification via Loss function Transformation
	(optional) Differentially Private Classification from Different Sources
Sep - Nov 2011	Privacy Preserving Speaker Identification/Verification
Dec 2011 - Mar 2012	Differentially Private HMM
	(optional) Privacy Preserving Graph Search
Apr - May 2012	Dissertation writing
Jun 2012	Defense

Bibliography

M. Atallah and J. Li. Secure outsourcing of sequence comparisons. *International Journal of Information Security*, 4 (4):277–287, 2005.

S. Avidan and M. Butman. Efficient methods for privacy preserving face detection. In *Neural Information Processing Systems*, pages 57–64, 2006.

B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Symposium on Principles of Database Systems*, pages 273–282, 2007.

D. Beaver. Foundations of secure interactive computing. In *CRYPTO*, pages 377–391, 1991.

M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

J. Benaloh. Dense Probabilistic Encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, Kingston, ON, Canada, May 1994.

F. Bimbot, J.-F. Bonastre, C. Fredouille, G. Gravier, I. Magrin-Chagnolleau, S. Meignier, T. Merlin, J. Ortega-Garcia, D. Petrovska-Delacretaz, and D. Reynolds. A tutorial on text-independent speaker verification. *EURASIP Journal on Applied Signal Processing*, 4:430–451, 2004.

A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The suLQ framework. In *Symposium on Principles of Database Systems*, 2005.

W. Campbell. Generalized linear discriminant sequence kernels for speaker recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2002.

W. Campbell, D. Sturim, and D. Reynolds. Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Processing Letters*, 13:308–311, 2006.

R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *Neural Information Processing Systems*, pages 289–296, 2008.

K. Chaudhuri, C. Monteleoni, and A. Sarwate. Differentially private empirical risk minimization. *arXiv:0912.0071v4 [cs.LG]*, 2010.

I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In 4^{th} *International Workshop on Practice and Theory in Public Key Cryptosystems*, pages 119–136, Cheju Island, Korea, Feb. 2001.

S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-28(4):357, 1980.

I. Dinur and K. Nissim. Revealing information while preserving privacy. In *ACM Symposium on Principles of Database Systems*, 2003.

C. Dwork. Differential privacy. In *International Colloquium on Automata, Languages and Programming*, 2006.

C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, 2004.

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

T. Ezzat and T. Poggio. Discriminative word-spotting using ordered spectro-temporal patch features. In *SAPA workshop at Interspeech*, 2008.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT*, pages 1–19. Springer, 2004.

O. Goldreich. *Foundations of cryptography: Volume II Basic Applications*. Cambridge University Press, 2004.

M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. `http://cvxr.com/cvx`, Apr. 2010.

A. Hatch and A. Stolcke. Generalized linear kernels for one-versus-all classification: Application to speaker recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006.

G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A practical differentially private random decision tree classifier. In *ICDM Workshop on Privacy Aspects of Data Mining*, pages 114–121, 2009.

M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2008.

M. Kearns, J. Tan, and J. Wortman. Privacy-preserving belief propagation and sampling. In *Neural Information Processing Systems*, 2007.

P. Kenny and P. Dumouchel. Experiments in speaker verification using factor analysis likelihood ratios. In *Odyssey*, pages 219–226, 2004.

X. Lin, C. Clifton, and M. Y. Zhu. Privacy-preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, 8(1):68–81, 2005.

Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):59–98, 2009.

P. C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2:49–55, 1936.

M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In *Advances in Cryptology: CRYPTO*, pages 573–590, Seattle, USA, 1999.

P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, second edition, 1984.

M. Pathak and B. Raj. Large margin multiclass Gaussian classification with differential privacy. In *ECML/PKDD Workshop on Privacy and Security issues in Data Mining and Machine Learning*, 2010a.

M. Pathak and B. Raj. Large margin Gaussian mixture models with differential privacy. In *IEEE Transactions on Dependable and Secure Computing (submitted)*, 2010b.

M. Pathak and B. Raj. Privacy-preserving protocols for eigenvector computation. In *ECML/PKDD Workshop on Privacy and Security issues in Data Mining and Machine Learning*, 2010c.

M. Pathak, S. Rane, and B. Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Neural Information Processing Systems*, 2010a.

M. Pathak, S. Rane, W. Sun, and B. Raj. Privacy preserving probabilistic inference with hidden markov models. In *International Conference on Acoustics, Speech and Signal Processing (submitted)*, 2010b.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, 1999.

M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard University, 1981.

L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-015157-2.

S. Rane and W. Sun. Privacy Preserving String Comparisons Based on Levenshtein Distance. In *Proc. IEEE international Workshop on Information Forensics and Security (WIFS)*, Seattle, USA, Dec. 2010.

D. Reynolds. An overview of automatic speaker recognition technology. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2002.

D. Reynolds, T. Quatieri, and R. Dunn. Speaker verification using adapted Gaussian mixture models. *Digital Signal Processing*, 10:19–41, 2000.

F. Sha and L. K. Saul. Large margin gaussian mixture modeling for phonetic classification and recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 265–268, 2006.

F. Sha and L. K. Saul. Large margin hidden markov models for automatic speech recognition. In *Neural Information Processing Systems*, pages 1249–1256, 2007.

P. Smaragdis and M. Shashanka. A framework for secure speech recognition. *IEEE Transactions on Audio, Speech & Language Processing*, 15(4):1404–1413, 2007.

K. Sridharan, S. Shalev-Shwartz, and N. Srebro. Fast rates for regularized objectives. In *Neural Information Processing Systems*, pages 1545–1552, 2008.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks, 2003.

J. Vaidya, C. Clifton, M. Kantarcioglu, and S. Patterson. Privacy-preserving decision trees over vertically partitioned data. *TKDD*, 2(3), 2008a.

J. Vaidya, M. Kantarcioglu, and C. Clifton. Privacy-preserving naive bayes classification. *VLDB J*, 17(4):879–898, 2008b.

J. Vaidya, H. Yu, and X. Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2): 161–178, 2008c.

A. Wang. An industrial strength audio search algorithm. In *International Conference on Music Information Retrieval*, 2003.

A. Yao. Protocols for secure computations (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, 1982.

Appendix A

Proofs of Theorems and Lemmas

A.1 Privacy Preserving Eigenvector Computation (Section 4.1.1)

proof of Theorem 4.1.1. We have,

$$\bar{M}^T\bar{M} = \begin{bmatrix} M^TM & M^TP \\ P^TM & I \end{bmatrix}.$$

Multiplying by the eigenvector $\bar{v} = \begin{bmatrix} v_{t\times 1} \\ v'_{s\times 1} \end{bmatrix}$ gives us

$$\bar{M}^T\bar{M}\begin{bmatrix} v \\ v' \end{bmatrix} = \begin{bmatrix} M^TMv + M^TPv' \\ P^TMv + v' \end{bmatrix} = \lambda\begin{bmatrix} v \\ v' \end{bmatrix}.$$

Therefore,

$$M^TMv + M^TPv' = \lambda v, \quad (A.1)$$

$$P^TMv + v' = \lambda v'. \quad (A.2)$$

Since $\lambda \neq 1$, Equation (A.2) implies $v' = \frac{1}{\lambda-1}P^TMv$. Substituting this into Equation (A.1) and the orthogonality of P gives us

$$M^TMv + \frac{1}{\lambda-1}M^TPP^TMv = \frac{\lambda}{\lambda-1}M^TMv = \lambda v.$$

Hence, $M^TMv = (\lambda - 1)v$. □

A.2 Differentially Private Large Margin Gaussian Mixture Models (Section 4.1.2)

Lemma A.2.1. *Assuming all the data instances to lie within a unit ℓ_2 ball, the difference in the derivative of Huber loss function $L(\mathbf{\Phi}, \vec{x}, y)$ calculated over two data instances $(\vec{x}_i, y_i)$ and $(\vec{x}'_i, y'_i)$ is bounded.*

$$\left\| \frac{\partial}{\partial \mathbf{\Phi}_{cm}} L(\mathbf{\Phi}, \vec{x}_i, y_i) - \frac{\partial}{\partial \mathbf{\Phi}_{cm}} L(\mathbf{\Phi}, \vec{x}'_i, y'_i) \right\| \;\leq\; 2.$$

Proof. For notational simplicity, let us denote the difference between the distance of The derivative of the Huber loss function for the data instance $\vec{x}_i$ with label y_i is

$$\frac{\partial}{\partial \mathbf{\Phi}_{cm}} L(\mathbf{\Phi}, \vec{x}_i, y_i) = \begin{cases} 0 & \text{if } M(x_i, \mathbf{\Phi}_c) > h, \\ \frac{-1}{2h}[h - \vec{x}_i^T \mathbf{\Phi}_{y_i} \vec{x}_i - \log \sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i}] \frac{e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i}}{\sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i}} \vec{x}_i \vec{x}_i^T & \text{if } |M(x_i, \mathbf{\Phi}_c)| \leq h, \\ \frac{e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i}}{\sum_m e^{-\vec{x}_i^T \mathbf{\Phi}_c \vec{x}_i}} \vec{x}_i \vec{x}_i^T & \text{if } M(x_i, \mathbf{\Phi}_c) < -h. \end{cases}$$

The data points lie in a ℓ_2 ball of radius 1, $\forall i : \|\vec{x}_i\|_2 \leq 1$. Using linear algebra, it is easy to show that the Frobenius norm of the matrix $\vec{x}_i\vec{x}_i^T$ is same as the ℓ_2 norm of the vector $\vec{x}_i$, $\|\vec{x}_i\vec{x}_i^T\| = \|\vec{x}_i\|_2 \leq 1$.

We have,

$$\left\| \frac{e^{-\vec{x}_i^T\mathbf{\Phi}_c\vec{x}_i}}{\sum_m e^{-\vec{x}_i^T\mathbf{\Phi}_c\vec{x}_i}}\vec{x}_i\vec{x}_i^T \right\| = \left\| \frac{1}{1+\sum_{m-1} e^{-\vec{x}_i^T\mathbf{\Phi}_c\vec{x}_i}}\vec{x}_i\vec{x}_i^T \right\| = \left| \frac{1}{1+\sum_{m-1} e^{-\vec{x}_i^T\mathbf{\Phi}_c\vec{x}_i}} \right| \|\vec{x}_i\vec{x}_i^T\| \leq 1.$$

The term $[h - \vec{x}_i^T\mathbf{\Phi}_{y_i}\vec{x}_i - \log\sum_m e^{-\vec{x}_i^T\mathbf{\Phi}_c\vec{x}_i}]$ is at most one when $|M(\vec{x}_i, \mathbf{\Phi}_c)| \leq h$, the Frobenius norm of the derivative of the Huber loss function is at most one in all cases, $\left\|\frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}, \vec{x}_i, y_i)\right\| \leq 1$. Similarly, for a data instance $\vec{x}_i'$ with label y_i', we have $\left\|\frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}, \vec{x}_i', y_i')\right\| \leq 1$.

Finally, using the triangle inequality $\|\vec{a} - \vec{b}\| = \|\vec{a} + (-\vec{b})\| \leq \|\vec{a}\| + \|\vec{b}\|$,

$$\left\| \frac{\partial}{\partial\mathbf{\Phi}_c}L(\mathbf{\Phi}, \vec{x}_i, y_i) - \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}, \vec{x}_i', y_i') \right\| \leq \left\| \frac{\partial}{\partial\mathbf{\Phi}_c}L(\mathbf{\Phi}, \vec{x}_i, y_i) \right\| + \left\| \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}, \vec{x}_i', y_i') \right\| \leq 2.$$

□

proof of Theorem 4.1.2. As $J(\mathbf{\Phi}, \vec{x}, \vec{y})$ is convex and differentiable, there is a unique solution $\mathbf{\Phi}^*$ that minimizes it. As the perturbation term $\sum_c\sum_{ij} b_{ij}\Phi_{cij}$ is also convex and differentiable, the perturbed objective function $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ also has a unique solution $\mathbf{\Phi}^p$ that minimizes it. Differentiating $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ wrt $\mathbf{\Phi}_{cm}$, we have

$$\frac{\partial}{\partial\mathbf{\Phi}_{cm}}J_p(\mathbf{\Phi}, \vec{x}, \vec{y}) = \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}, \vec{x}, \vec{y}) + \lambda\mathbf{I}_{\mathbf{\Phi}} + \mathbf{b}. \tag{A.3}$$

Substituting the optimal $\mathbf{\Phi}_{cm}^p$ in the derivative gives us

$$\lambda\mathbf{I}_{\mathbf{\Phi}} + \mathbf{b} = -\frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}, \vec{y}).$$

This relation shows that two different values of $\mathbf{b}$ cannot result in the same optimal $\mathbf{\Phi}^p$. As the perturbed objective function $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ is also convex and differentiable, there is a bijective map between the perturbation $\mathbf{b}$ and the unique $\mathbf{\Phi}^p$ minimizing $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$.

Let $\mathbf{b_1}$ and $\mathbf{b_2}$ be the two perturbations applied when training with the adjacent datasets $(\vec{x}, \vec{y})$ and $(\vec{x}', \vec{y}')$, respectively. Assuming that we obtain the same optimal solution $\mathbf{\Phi}^p$ while minimizing both $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ with perturbation $\mathbf{b}_1$ and $J_p(\mathbf{\Phi}, \vec{x}, \vec{y})$ with perturbation $\mathbf{b}_2$,

$$\begin{aligned}
\lambda\mathbf{I}_{\mathbf{\Phi}} + \mathbf{b}_1 &= -\frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}, \vec{y}),\\
\lambda\mathbf{I}_{\mathbf{\Phi}} + \mathbf{b}_2 &= -\frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}', \vec{y}'),\\
\mathbf{b}_1 - \mathbf{b}_2 &= \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}', \vec{y}') - \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}, \vec{y}). \qquad \text{(A.4)}
\end{aligned}$$

We take the Frobenius norm of both sides and apply the bound on the the RHS as given by Lemma A.2.1.

$$\begin{aligned}
\|\mathbf{b}_1 - \mathbf{b}_2\| &= \left\| \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}', \vec{y}') - \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}, \vec{y}) \right\|\\
&= \Bigg\| \sum_{i=1}^{n-1} \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}_i, y_i) + \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}_n', y_n')\\
&\qquad - \sum_{i=1}^{n-1} \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}_i, y_i) - \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}_n, y_n) \Bigg\|\\
&= \left\| \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}_n', y_n') - \frac{\partial}{\partial\mathbf{\Phi}_{cm}}L(\mathbf{\Phi}^p, \vec{x}_n, y_n) \right\| \leq 2.
\end{aligned}$$

Using this property, we can calculate the ratio of densities of drawing the perturbation matrices $\mathbf{b}_1$ and $\mathbf{b}_2$ as

$$\frac{P(\mathbf{b}=\mathbf{b}_1)}{P(\mathbf{b}=\mathbf{b}_2)} = \frac{\frac{1}{\text{surf}(\|\mathbf{b}_1\|)}\|\mathbf{b}_1\|^d \exp\left[-\frac{\epsilon}{2}\|\mathbf{b}_1\|\right]}{\frac{1}{\text{surf}(\|\mathbf{b}_2\|)}\|\mathbf{b}_2\|^d \exp\left[-\frac{\epsilon}{2}\|\mathbf{b}_2\|\right]},$$

where surf($\|\mathbf{b}\|$) is the surface area of the $(d+1)$-dimensional hypersphere with radius $\|\mathbf{b}\|$. As $\text{surf}(\|\mathbf{b}\|) = \text{surf}(1)\|\mathbf{b}\|^d$, where surf$(1)$ is the area of the unit $(d+1)$-dimensional hypersphere, the ratio of the densities becomes

$$\begin{aligned}\frac{P(\mathbf{b}=\mathbf{b}_1)}{P(\mathbf{b}=\mathbf{b}_2)} &= \exp\left[\frac{\epsilon}{2}(\|\mathbf{b}_2\| - \|\mathbf{b}_1\|)\right] \\ &\le \exp\left[\frac{\epsilon}{2}\|\mathbf{b}_2 - \mathbf{b}_1\|\right] \le \exp(\epsilon). \end{aligned} \tag{A.5}$$

The ratio of the densities of learning $\mathbf{\Phi}^p$ using the adjacent datasets $(\vec{x}, \vec{y})$ and $(\vec{x}', \vec{y}')$ is given by

$$\frac{P(\mathbf{\Phi}^p|\vec{x},\vec{y})}{P(\mathbf{\Phi}^p|\vec{x}',\vec{y}')} = \frac{P(\mathbf{b}=\mathbf{b}_1)}{P(\mathbf{b}=\mathbf{b}_2)} \frac{|\det(\mathbf{J}(\mathbf{\Phi}^p \to \mathbf{b}_1|\vec{x},\vec{y}))|^{-1}}{|\det(\mathbf{J}(\mathbf{\Phi}^p \to \mathbf{b}_2|\vec{x}',\vec{y}'))|^{-1}}, \tag{A.6}$$

where $\mathbf{J}(\mathbf{\Phi}^p \to \mathbf{b}_1|\vec{x},\vec{y})$ and $\mathbf{J}(\mathbf{\Phi}^p \to \mathbf{b}_2|\vec{x}',\vec{y}')$ are the Jacobian matrices of the bijective mappings from $\mathbf{\Phi}^p$ to $\mathbf{b}_1$ and $\mathbf{b}_2$, respectively. Following a procedure identical to Theorem 2 of Chaudhuri et al. [2010] (omitted due to lack of space), it can be shown that the ratio of Jacobian determinants is upper bounded by a constant factor $\exp(k) = 1 + \frac{2\alpha}{n\lambda} + \frac{\alpha^2}{n^2\lambda^2}$ for a constant value of α. Therefore, the ratio of the densities of learning $\mathbf{\Phi}^p$ using the adjacent datasets becomes

$$\frac{P(\mathbf{\Phi}^p|\vec{x},\vec{y})}{P(\mathbf{\Phi}^p|\vec{x}',\vec{y}')} \le \exp(\epsilon + k) = \exp(\epsilon'). \tag{A.7}$$

Similarly, we can show that the probability ratio is lower bounded by $\exp(-\epsilon')$, which together with Equation (A.7) satisfies the definition of differential privacy. □

Lemma A.2.2. *The objective function $J(\mathbf{\Phi})$ is λ-strongly convex. For $0 \le \alpha \le 1$,*

$$\begin{aligned} J\left(\alpha\mathbf{\Phi} + (1-\alpha)\mathbf{\Phi}'\right) &\le \alpha J(\mathbf{\Phi}) + (1-\alpha)J(\mathbf{\Phi}') \\ &\quad - \frac{\lambda\alpha(1-\alpha)}{2}\sum_c \|\mathbf{\Phi}_c - \mathbf{\Phi}'_c\|^2. \end{aligned}$$

Proof. By definition, Huber loss is λ-strongly convex, *i.e.*,

$$\begin{aligned} L\left(\alpha\mathbf{\Phi} + (1-\alpha)\mathbf{\Phi}'\right) &\le \alpha L(\mathbf{\Phi}) + (1-\alpha)L(\mathbf{\Phi}') \\ &\quad - \frac{\lambda\alpha(1-\alpha)}{2}\|\mathbf{\Phi} - \mathbf{\Phi}'\|^2. \end{aligned} \tag{A.8}$$

where the Frobenius norm of the matrix set $\mathbf{\Phi} - \mathbf{\Phi}'$ is the sum of norms of the component matrices $\mathbf{\Phi}_c - \mathbf{\Phi}'_c$,

$$\|\mathbf{\Phi} - \mathbf{\Phi}'\|^2 = \sum_c \|\mathbf{\Phi}_c - \mathbf{\Phi}'_c\|^2. \tag{A.9}$$

As the regularization term $N(\mathbf{\Phi})$ is linear,

$$\begin{aligned} &N(\alpha\mathbf{\Phi} + (1-\alpha)\mathbf{\Phi}') \\ &= \lambda\sum_{cm}\text{trace}(\alpha\mathbf{I}_{\mathbf{\Phi}}\mathbf{\Phi}_{cm}\mathbf{I}_{\mathbf{\Phi}} + (1-\alpha)\mathbf{I}_{\mathbf{\Phi}}\mathbf{\Phi}'_{cm}\mathbf{I}_{\mathbf{\Phi}}) \\ &= \alpha\lambda\sum_{cm}\text{trace}(\mathbf{I}_{\mathbf{\Phi}}\mathbf{\Phi}_{cm}\mathbf{I}_{\mathbf{\Phi}}) + (1-\alpha)\lambda\sum_{cm}\text{trace}(\mathbf{I}_{\mathbf{\Phi}}\mathbf{\Phi}'_{cm}\mathbf{I}_{\mathbf{\Phi}}) \\ &= \alpha N(\mathbf{\Phi}) + (1-\alpha)N(\mathbf{\Phi}'). \end{aligned} \tag{A.10}$$

The lemma follows directly from the definition $J(\mathbf{\Phi}) = L(\mathbf{\Phi}) + N(\mathbf{\Phi})$.

□

Lemma A.2.3.

$$\frac{1}{dC}\left[\sum_c \mathit{trace}[\mathbf{I}_{\boldsymbol{\Phi}}(\boldsymbol{\Phi}_c - \boldsymbol{\Phi}'_c)\mathbf{I}_{\boldsymbol{\Phi}}]\right]^2 \le \sum_c \|\boldsymbol{\Phi}_c - \boldsymbol{\Phi}'_c\|^2 .$$

Proof. Let $\Phi_{c,i,j}$ be the $(i,j)^{\text{th}}$ element of the size $(d+1)\times(d+1)$ matrix $\boldsymbol{\Phi}_c - \boldsymbol{\Phi}'_c$. By the definition of the Frobenius norm, and using the identity $N\sum_{i=1}^N x_i^2 \ge (\sum_{i=1}^N x_i)^2$,

$$\begin{aligned}\sum_c \|\boldsymbol{\Phi}_c - \boldsymbol{\Phi}'_c\|^2 &= \sum_c \sum_{i=1}^{d+1}\sum_{j=1}^{d+1} \Phi_{c,i,j}^2 \ge \sum_c \sum_{i=1}^{d+1} \Phi_{c,i,i}^2 \\ &\ge \sum_c \sum_{i=1}^{d} \Phi_{c,i,i}^2 \ge \frac{1}{dC}\left(\sum_c \sum_{i=1}^{d} \Phi_{c,i,i}\right)^2 \\ &= \frac{1}{dC}\left[\sum_c \text{trace}[\mathbf{I}_{\boldsymbol{\Phi}}(\boldsymbol{\Phi}_c - \boldsymbol{\Phi}'_c)\mathbf{I}_{\boldsymbol{\Phi}}]\right]^2 .\end{aligned}$$

□

Lemma A.2.4.

$$P\left[\|\mathbf{b}\| \ge \frac{2(d+1)^2}{\epsilon}\log\left(\frac{d}{\delta}\right)\right] \le \delta.$$

Proof. Similar to the union bound argument used in Lemma 5 in Chaudhuri and Monteleoni [2008].

□

A.3 Differentially Private Classification from Multiparty Data (Section 4.1.3)

Theorem A.3.1. *Given a set of n training data instances lying in a ball of radius 1, the sensitivity of regularized logistic regression classifier is at most $\frac{2}{n\lambda}$. If $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are classifiers trained on adjacent datasets of size n with regularization parameter λ,*

$$\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|_1 \le \frac{2}{n\lambda}.$$

Lemma A.3.2. *Let $G(\boldsymbol{w})$ and $g(\boldsymbol{w})$ be two differentiable, convex functions of $\boldsymbol{w}$. If $\boldsymbol{w}_1 = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$ and $\boldsymbol{w}_2 = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w}) + g(\boldsymbol{w})$, then $\|\boldsymbol{w}_1 - \boldsymbol{w}_2\| \le \frac{g_1}{G_2}$ where $g_1 = \max_{\boldsymbol{w}} \|\nabla g(\boldsymbol{w})\|$ and $G_2 = \min_{\boldsymbol{v}} \min_{\boldsymbol{w}} \boldsymbol{v}^T \nabla^2 G(\boldsymbol{w})\boldsymbol{v}$ for any unit vector $\boldsymbol{v} \in \mathbb{R}^d$.*

Proof of Theorem 4.1.6. We formulate the problem of estimating the individual classifiers $\hat{\mathbf{w}}_j$ and the classifier $\mathbf{w}^*$ trained over the entire training data in terms of minimizing the two differentiable and convex functions $g(\mathbf{w})$ and $G(\mathbf{w})$.

$$\begin{aligned}\hat{\mathbf{w}}_j &= \underset{w}{\text{argmin}}\ J(\mathbf{w}, \mathbf{x}|_j, \mathbf{y}|_j) = \underset{w}{\text{argmin}}\ G(\mathbf{w}), \\ \mathbf{w}^* &= \underset{w}{\text{argmin}}\ J(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \underset{w}{\text{argmin}}\ L(\mathbf{w}, \mathbf{x}|_j, \mathbf{y}|_j) + \sum_{l\in[K]-j} L(\mathbf{w}, \mathbf{x}|_l, \mathbf{y}|_l) + \lambda\|\mathbf{w}\|^2 \\ &= \underset{w}{\text{argmin}}\ J(\mathbf{w}, \mathbf{x}|_j, \mathbf{y}|_j) + \sum_{l\in[K]-j} L(\mathbf{w}, \mathbf{x}|_l, \mathbf{y}|_l) = \underset{w}{\text{argmin}}\ G(\mathbf{w}) + g(\mathbf{w}).\end{aligned}$$

$$\nabla g(\mathbf{w}) = \sum_{l\in[K]-j} \frac{1}{n_l} \sum_{i=1}^{n_l} \frac{-e^{-y_i|_l w^T x_i|_l}}{1+e^{-y_i|_l w^T x_i|_l}} y_i x_i^T,$$

$$\|\nabla g(\mathbf{w})\| = \left\| \sum_l \frac{1}{n_l} \sum_i \frac{-e^{-y_i|_l w^T x_i|_l}}{1+e^{-y_i|_l w^T x_i|_l}} y_i x_i^T \right\| \le \sum_l \frac{1}{n_l} \left\| \sum_i \frac{-e^{-y_i|_l w^T x_i|_l}}{1+e^{-y_i|_l w^T x_i|_l}} y_i x_i^T \right\| \le \sum_l \frac{1}{n_l},$$

$$g_1 = \max_{\mathbf{w}} \|\nabla g(\mathbf{w})\| \le \sum_l \frac{1}{n_l}. \tag{A.11}$$

$$\nabla^2 G(\mathbf{w}) = \frac{1}{n_j} \sum_i \frac{e^{y_i|_j \mathbf{w}^T \mathbf{x}_i|_j}}{1+e^{y_i|_j \mathbf{w}^T \mathbf{x}_i|_j}} \mathbf{x}_i|_j \mathbf{x}_i|_j^T + \lambda \mathbf{1}, \;\; G_2 \ge \lambda. \tag{A.12}$$

Substituting the bounds on g_1 and G_2 in Lemma A.3.2,

$$\|\hat{\mathbf{w}}_j - \mathbf{w}^*\| \le \frac{1}{\lambda} \sum_l \frac{1}{n_l}. \tag{A.13}$$

Applying triangle inequality,

$$\|\hat{\mathbf{w}} - \mathbf{w}^*\| = \left\| \frac{1}{K} \sum_j \hat{\mathbf{w}}_j - \mathbf{w}^* \right\| = \frac{1}{K} \left\| \sum_j \hat{\mathbf{w}}_j - K\mathbf{w}^* \right\| = \frac{1}{K} \|(\hat{\mathbf{w}}_1 - \mathbf{w}^*) + \ldots + (\hat{\mathbf{w}}_K - \mathbf{w}^*)\|$$

$$\le \frac{1}{K} \sum_j \|\hat{\mathbf{w}}_j - \mathbf{w}^*\| = \frac{1}{K\lambda} \sum_j \sum_{l\in[K]-j} \frac{1}{n_l} = \frac{K-1}{K\lambda} \sum_j \frac{1}{n_j} \le \frac{K-1}{n_{(1)}\lambda}.$$

where $n_{(1)} = \min_j n_j$. □

proof of Theorem 4.1.5. Consider the case where one instance of the training dataset D is changed to result in an adjacent dataset D'. This would imply a change in one element in the training dataset of one party and thereby a change in the corresponding learned vector $\hat{\mathbf{w}}_j^s$. Assuming that the change is in the dataset of the party P_j, the change in the learned vector is only going to be in $\hat{\mathbf{w}}_j$; let denote the new classifier by $\hat{\mathbf{w}}_j'$. In Theorem A.3.1, we bound the sensitivity of $\hat{\mathbf{w}}_j$ as $\|\hat{\mathbf{w}}_j - \hat{\mathbf{w}}_j'\|_1 \le \frac{2}{n_j \epsilon \lambda}$. Following an argument similar to [Dwork et al., 2006], considering that we learn the same vector $\hat{\mathbf{w}}^s$ using either the training datasets D and D', we have

$$\frac{P\left(\hat{\mathbf{w}}^s|D\right)}{P\left(\hat{\mathbf{w}}^s|D'\right)} = \frac{P\left(\hat{\mathbf{w}}_j + \boldsymbol{\eta}|D\right)}{P\left(\hat{\mathbf{w}}_j' + \boldsymbol{\eta}|D'\right)} = \frac{\exp\left[\frac{n_{(1)}\epsilon\lambda}{2}\|\hat{\mathbf{w}}_j\|_1\right]}{\exp\left[\frac{n_{(1)}\epsilon\lambda}{2}\|\hat{\mathbf{w}}_j'\|_1\right]} \le \exp\left[\frac{n_{(1)}\epsilon\lambda}{2}\|\hat{\mathbf{w}}_j - \hat{\mathbf{w}}_j'\|_1\right]$$

$$\le \exp\left[\frac{n_{(1)}\epsilon\lambda}{2}\frac{2}{n_j\lambda}\right] \le \exp\left[\frac{n_{(1)}}{n_j}\epsilon\right] \le \exp(\epsilon),$$

by the definition of function sensitivity. Similarly, we can lower bound the the ratio by $\exp(-\epsilon)$. □

Proof of Theorem 4.1.3. We use the Taylor series expansion of the function J to have

$$J(\hat{\mathbf{w}}^s) = J(\mathbf{w}^*) + (\hat{\mathbf{w}}^s - \mathbf{w}^*)^T \nabla J(\mathbf{w}^*) + \frac{1}{2}(\hat{\mathbf{w}}^s - \mathbf{w}^*)^T \nabla^2 J(\mathbf{w})(\hat{\mathbf{w}}^s - \mathbf{w}^*)$$

for some $\mathbf{w} \in \mathbb{R}^d$. By definition, $\nabla J(\mathbf{w}^*) = 0$.

Taking ℓ_2 norm on both sides and applying Cauchy-Schwarz inequality,

$$|J(\hat{\mathbf{w}}^s) - J(\mathbf{w}^*)| \le \frac{1}{2}\|\hat{\mathbf{w}}^s - \mathbf{w}^*\|^2 \; \|\nabla^2 J(\mathbf{w})\|. \tag{A.14}$$

The second gradient of the regularized loss function for logistic regression is

$$\nabla^2 J(\mathbf{w}) = \frac{1}{n}\sum_i \frac{e^{y_i \mathbf{w}^T \mathbf{x}_i}}{1+e^{y_i \mathbf{w}^T \mathbf{x}_i}} \mathbf{x}_i \mathbf{x}_i^T + \lambda \mathbf{1} = \frac{1}{n}\sum_i \frac{1}{1+e^{-y_i \mathbf{w}^T \mathbf{x}_i}} \mathbf{x}_i \mathbf{x}_i^T + \lambda \mathbf{1}.$$

Since the logistic function term is always less than one and all $\mathbf{x}_i$ lie in a unit ball, $\|\nabla^2 J(\mathbf{w})\| \le \lambda + 1$. Substituting this into Equation A.14 and using the fact that $J(\mathbf{w}^*) \le J(\mathbf{w}), \forall \mathbf{w} \in \mathbb{R}^d$,

$$J(\hat{\mathbf{w}}^s) \le J(\mathbf{w}^*) + \frac{\lambda+1}{2}\|\hat{\mathbf{w}}^s - \mathbf{w}^*\|^2. \tag{A.15}$$

The classifier $\hat{\mathbf{w}}^s$ is the perturbed aggregate classifier, *i.e.*, $\hat{\mathbf{w}}^s = \hat{\mathbf{w}} + \boldsymbol{\eta}$, with the noise term $\boldsymbol{\eta} \sim \text{Lap}\left(\frac{2}{n_{(1)}\epsilon\lambda}\right)$. We apply Lemma A.3.3 to bound $\|\boldsymbol{\eta}\|$ with probability at least $1-\delta$. Substituting this into Equation A.15, we have

$$\begin{aligned} J(\hat{\mathbf{w}}^s) &\le J(\mathbf{w}^*) + \frac{1}{2}\|\hat{\mathbf{w}} - \mathbf{w}^* + \boldsymbol{\eta}\|^2 = J(\mathbf{w}^*) + \frac{\lambda+1}{2}\left[\|\hat{\mathbf{w}} - \mathbf{w}^*\|^2 + \|\boldsymbol{\eta}\|^2 + 2(\hat{\mathbf{w}} - \mathbf{w}^*)^T\boldsymbol{\eta}\right] \\ &\le J(\mathbf{w}^*) + \frac{\lambda+1}{2}\left[\frac{(K-1)^2}{n_{(1)}^2\lambda^2} + \frac{4d^2}{n_{(1)}^2\epsilon^2\lambda^2}\log^2\left(\frac{d}{\delta}\right)\right] + (\lambda+1)|(\hat{\mathbf{w}} - \mathbf{w}^*)^T\boldsymbol{\eta}|. \end{aligned}$$

Using the Cauchy-Schwarz inequality on the last term,

$$\begin{aligned} J(\hat{\mathbf{w}}^s) &\le J(\mathbf{w}^*) + \frac{\lambda+1}{2}\left[\frac{(K-1)^2}{n_{(1)}^2\lambda^2} + \frac{4d^2}{n_{(1)}^2\epsilon^2\lambda^2}\log^2\left(\frac{d}{\delta}\right)\right] + (\lambda+1)\|\hat{\mathbf{w}} - \mathbf{w}^*\|\|\boldsymbol{\eta}\| \\ &\le J(\mathbf{w}^*) + \frac{(K-1)^2(\lambda+1)}{2n_{(1)}^2\lambda^2} + \frac{2d^2(\lambda+1)}{n_{(1)}^2\epsilon^2\lambda^2}\log^2\left(\frac{d}{\delta}\right) + \frac{2d(K-1)(\lambda+1)}{n_{(1)}^2\epsilon\lambda^2}\log\left(\frac{d}{\delta}\right). \end{aligned}$$

□

Proof of Theorem 4.1.8. Let $\mathbf{w}^r$ be the classifier minimizing the true risk $\tilde{J}(\mathbf{w})$. By rearranging the terms,

$$\tilde{J}(\hat{\mathbf{w}}^s) = \tilde{J}(\mathbf{w}^*) + [\tilde{J}(\hat{\mathbf{w}}^s) - \tilde{J}(\mathbf{w}^r)] + [\tilde{J}(\mathbf{w}^r) - \tilde{J}(\mathbf{w}^*)] \le \tilde{J}(\mathbf{w}^*) + [\tilde{J}(\hat{\mathbf{w}}^s) - \tilde{J}(\mathbf{w}^r)].$$

Sridharan, et al. Sridharan et al. [2008] present a bound between the true excess risk of any classifier as an expression of bound on the regularized empirical risk for that classifier and the classifier minimizing the regularized empirical risk. With probability at least $1-\delta$,

$$\tilde{J}(\hat{\mathbf{w}}^s) - \tilde{J}(\mathbf{w}^r) \le 2[J(\hat{\mathbf{w}}^s) - J(\mathbf{w}^*)] + \frac{16}{\lambda n}\left[32 + \log\left(\frac{1}{\delta}\right)\right]. \tag{A.16}$$

Substituting the bound from Theorem 4.1.3,

$$\tilde{J}(\hat{\mathbf{w}}^s) - \tilde{J}(\mathbf{w}^r) \le \frac{2(K-1)^2(\lambda+1)}{2n_{(1)}^2\lambda^2} + \frac{4d^2(\lambda+1)}{n_{(1)}^2\epsilon^2\lambda^2}\log^2\left(\frac{d}{\delta}\right) \tag{A.17}$$

$$+ \frac{4d(K-1)(\lambda+1)}{n_{(1)}^2\epsilon\lambda^2}\log\left(\frac{d}{\delta}\right) + \frac{16}{\lambda n}\left[32 + \log\left(\frac{1}{\delta}\right)\right]. \tag{A.18}$$

Substituting this bound into Equation A.16 gives us a bound on the true excess risk of the classifier $\hat{\mathbf{w}}^s$ over the classifier $\mathbf{w}^*$. □

Lemma A.3.3. *Given a d-dimensional random variable $\boldsymbol{\eta} \sim Lap(\beta)$* i.e., $P(\boldsymbol{\eta}) = \frac{1}{2\beta}e^{-\frac{\|\boldsymbol{\eta}\|_1}{\beta}}$, *with probability at least $1-\delta$, the ℓ_2 norm of the random variable is bounded as*

$$\|\boldsymbol{\eta}\| \le 2d\beta\log\left(\frac{d}{\delta}\right).$$

The proof is similar to Lemma 5 of Chaudhuri and Monteleoni [2008].