Data Sparseness in Linear SVM

Xiang Li[*†], **Huaimin Wang**[†], **Bin Gu**[*‡], **Charles X. Ling**[*1]
[*]Computer Science Department, University of Western Ontario, Canada
[†]School of Computer, National University of Defense Technology, China
[‡]Nanjing University of Information Science and Technology, China
lxiang2@uwo.ca, hmwang@nudt.edu.cn, jsgubin@nuist.edu.cn, cling@csd.uwo.ca

Abstract

Large sparse datasets are common in many real-world applications. Linear SVM has been shown to be very efficient for classifying such datasets. However, it is still unknown how data sparseness would affect its convergence behavior. To study this problem in a systematic manner, we propose a novel approach to generate large and sparse data from real-world datasets, using statistical inference and the data sampling process in the PAC framework. We first study the convergence behavior of linear SVM experimentally, and make several observations, useful for real-world applications. We then offer theoretical proofs for our observations by studying the Bayes risk and PAC bound. Our experiment and theoretic results are valuable for learning large sparse datasets with linear SVM.

1 Introduction

Large sparse datasets are common in many real-world applications. They contain millions of data instances and attributes, but most of the attribute values are missing or unobserved. A good example is the user-item data of various popular on-line services. Netflix has published part of its movie rating dataset [Bennett and Lanning, 2007] for algorithm competition. This data consists of 100,480,507 ratings given by 480,189 users to 17,770 movies, which amounts to a sparseness of 98.822%. Data sparseness becomes even higher in other domains. For example, the Flickr dataset collected by [Cha *et al.*, 2009] contains the 'favorite' marks given by 497,470 users on 11,195,144 photos, its sparseness reaches 99.9994%. Such high sparseness is understandable: compared to the massive amount of available items, each user could only have consumed or interacted with a tiny portion of them. In many situations, we need to classify these datasets to make predictions. For example, in personalized recommender systems and targeted advertising, we would like to predict various useful features of a user (gender, income level, location and so on) based on items she/he has visited.

[1]Charles X. Ling (cling@csd.uwo.ca) is the corresponding author.

Linear SVM such as Pegasos [Shalev-Shwartz *et al.*, 2011] and LibLinear [Fan *et al.*, 2008] are popular choices to classify large sparse datasets efficiently, because they scale linearly with the number of non-missing values. Nowadays, it is possible to train linear SVM on very large datasets. Theoretically, it is known that larger training data will lead to lower generalization error, and asymptotically it will converge to the lowest error that can be achieved [Vapnik, 1999]. However, it is still hard to answer the following important questions about linear SVM and data sparseness:

(1). If we put in effort to reduce data sparseness, would it decrease the asymptotic generalization error of linear SVM?

(2). Would data sparseness affect the amount of training data needed to approach the asymptotic generalization error of linear SVM?

These questions essentially concern the convergence behavior of learning, which has been addressed in previous works. In Statistical Learning Theory [Vapnik, 1999], PAC bound gives high-probability guarantee on the convergence between training and generalization error. Once the VC-dimension of the problem is known, PAC bound could predict the amount of training instances needed to approach the asymptotic error. Other works [Bartlett and Shawe-Taylor, 1999] have shown that the VC-dimension of linear SVM is closely related to the hyperplane margin over the data space. As an initial step of understanding the impact of data sparseness on the margin of hyperplanes, a recent work [Long and Servedio, 2013] has given bounds for integer weights of separating hyperplanes over the k-sparse Hamming Ball space $\mathbf{x} \in \{0,1\}^m_{\leq k}$ (at most k of the m attributes are non-zero). There also exist several SVM variants that could deal with missing data [Pelckmans *et al.*, 2005; Chechik *et al.*, 2008]. However, still no previous work could explicitly predict the convergence behavior of linear SVM when data is highly sparse.

In this paper, we will answer this question by systematic experiments and then verify our findings through theoretic study. We propose a novel approach to generate large and sparse synthetic data from real-world datasets. First, we infer the statistical distribution of real-world movie rating datasets using a recent Probabilistic Matrix Factorization (PMF) inference algorithm [Lobato *et al.*, 2014]. From the inferred distribution, we then sample a large number of data instances following the PAC framework, so we can study the general-

ization error with various training sizes.

In order to study the effect of data sparseness, we consider a simple data missing model, which allows us to systematically vary the degree of data sparseness and compare the learning curves of linear SVM. To follow the PAC framework, we study the curves of training and testing error rates as we keep increasing the training size. We have made several important observations about how data sparseness would affect the asymptotic generalization error and the rate of convergence when using linear SVM; see Section 3.

We then analyze our observations with detailed theoretic study. For asymptotic generalization error, we study the change of Bayes risk as we vary data sparseness. With proper assumptions, we have proved that higher sparseness will increase the Bayes risk of the data; see Section 4.1. For the asymptotic rate of convergence, we observe that different sparseness would not change the amount of training data needed to approach convergence. We study its theoretic reason using the PAC bound; see Section 4.2.

Our results are very useful for using linear SVM in real-world applications. Firstly, they indicate that sparser data would generally increase the asymptotic error, which encourages practitioners to put effort in reducing data sparseness. Secondly, our results also imply that sparser data will not lead to slower learning curve convergence when using linear SVM, although the asymptotic generalization error rate would increase.

The rest of the paper is organized as follows. Section 2 gives details of our data generation approach. Section 3 describes our experiment and observations. Section 4 proves our observations. The final section concludes the paper.

2 A Novel Approach to Generate Sparse Data

To systematically study the impact of data sparseness on linear SVM, in this section, we first propose a novel approach to generate sparse data.

2.1 Basic Settings

To simplify the problem setting, we only consider binary attribute values $\mathbf{x} \in \{0,1\}^m$ and binary label $y \in \{+,-\}$. Our data generation approach follows the standard setting of the PAC framework [Vapnik, 1999], which assumes that data $(\mathbf{x}, y)$ are independently sampled from a fixed distribution $P(\mathbf{x}, y) = P(y)P(\mathbf{x}|y)$. In addition, our approach uses distribution inferred from real-world datasets, so the generated data are realistic in a statistical sense. Specifically, we use datasets of recommendation systems such as movie ratings to infer $P(\mathbf{x}, y)$.

Recently, datasets of recommendation systems are widely studied. They usually contain ordinal ratings (e.g., 0 to 5) given by each user to each item. We consider each user as a data instance and its rating on items corresponds to each attribute. To make the data binary, we only consider the presence/absence of each rating. We choose the gender of each user as its label $y \in \{+,-\}$.

To infer the distribution $P(\mathbf{x}, y)$, we use Probabilistic Matrix Factorization [Mnih and Salakhutdinov, 2007] which is a widely-used probabilistic modeling framework for user-item ratings data. In the next subsection, we will briefly review the PMF model for binary datasets proposed by [Lobato *et al.*, 2014], which will be used in our data generation process.

2.2 Review of PMF for Sparse Binary Data

Consider L users and M items, the $L \times M$ binary matrix $\mathbf{X}$ indicates whether each rating exists. In this case, $\mathbf{X}$ could be modeled by the PMF given in [Lobato *et al.*, 2014] which has good predictive performance:

$$p(\mathbf{X}|\mathbf{U}, \mathbf{V}, z) = \prod_{i=1}^{L} \prod_{j=1}^{M} p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) \tag{1}$$

where a Bernoulli likelihood is used along with the Matrix Factorization assumption $\mathbf{X} \approx \mathbf{U} \cdot \mathbf{V}$ to model each binary entry of $\mathbf{X}$:

$$p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) = Ber(x_{i,j}|\delta(\mathbf{u}_i \mathbf{v}_j^T + z)) \tag{2}$$

where $\delta(\cdot)$ is the logistic function $\delta(x) = \frac{1}{1+exp(-x)}$ to squash a real number into the range of $(0, 1)$, $Ber(\cdot)$ is the pdf of a Bernoulli distribution and z is a global bias parameter in order to handle data sparsity.

This PMF model further assumes that all latent variables are independent by using fully factorized Gaussian priors:

$$p(\mathbf{U}) = \prod_{i=1}^{L} \prod_{d=1}^{D} \mathcal{N}(u_{i,d}|m_{i,d}^u, v_{i,d}^u), \tag{3}$$

$$p(\mathbf{V}) = \prod_{j=1}^{M} \prod_{d=1}^{D} \mathcal{N}(v_{j,d}|m_{i,d}^v, v_{i,d}^v), \tag{4}$$

$$p(z) = \mathcal{N}(z|m^z, v^z) \tag{5}$$

where $\mathcal{N}(\cdot|m, v)$ denotes the pdf of a Gaussian distribution with mean m and variance v. Given a binary dataset $\mathbf{X}$, the model posterior $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X})$ could be inferred using the SIBM algorithm described in [Lobato *et al.*, 2014]. The posterior predictive distributions are used for predicting each binary value in $\mathbf{X}$:

$$p(\widetilde{\mathbf{X}}|\mathbf{X}) = \int p(\widetilde{\mathbf{X}}|\mathbf{U}, \mathbf{V}, z) p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}) d\mathbf{U} d\mathbf{V} dz \tag{6}$$

2.3 The Distribution for Data Sampling

Based on the above PMF model, we next describe the distribution for sampling synthetic data. Since the distribution is inferred from a given real-world dataset $\mathbf{X}$, we denote it as

$$p(\widetilde{\mathbf{x}}, y|\mathbf{X}) = p(y)p(\widetilde{\mathbf{x}}|y, \mathbf{X}) \tag{7}$$

For $p(y)$, we simply use a balanced class prior, $p(+) = p(-) = 0.5$. This ensures equal chances of getting positive and negative instances when sampling.

Now we describe how we get $p(\widetilde{\mathbf{x}}|y, \mathbf{X})$. We first divide the real-world dataset $\mathbf{X}$ into the positive labeled set $\mathbf{X}_+$ and the set of negative instances, $\mathbf{X}_-$. We infer the model posteriors $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_-)$ and $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_+)$ separately using the SIBM algorithm.

Notice that these inferred models can not be directly used to generate infinite samples: suppose the number of users in $\mathbf{X}_+$ (and $\mathbf{X}_-$) is L_+ (and L_-), the posterior predictive (6) only gives probabilities for each of these L_+ (L_-) existing users. In other words, the predictive distribution could only be used to reconstruct or predict the original $\mathbf{X}$, as did in the original paper [Lobato *et al.*, 2014].

In order to build a distribution for sampling an infinite number of synthetic users, we employ the following stochastic process: whenever we need to sample a new synthetic instance $\widetilde{\mathbf{x}}$, we first randomly choose a user i from the L_+ (or L_-) existing users, and we then sample the synthetic instance using the posterior predictive of this user. Using this process, the pdf of $p(\widetilde{\mathbf{x}}|y, \mathbf{X})$ is actually

$$p(\widetilde{\mathbf{x}}|y, \mathbf{X}) = \sum_{i=1}^{L_y} p(i, y) \cdot \int p(\widetilde{\mathbf{x}}|\mathbf{U}, \mathbf{V}, z, i) p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_y) d\mathbf{U} d\mathbf{V} dz \quad (8)$$

where $y \in \{+, -\}$, $p(i, y) = \frac{1}{L_y}$ and

$$p(\widetilde{\mathbf{x}}|\mathbf{U}, \mathbf{V}, z, i) = \prod_{j=1}^{M} p(\widetilde{x}_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) \quad (9)$$

is the likelihood for each instance (existing user) of $\mathbf{X}_y$, which is equivalent to (1). The process of sampling data from $p(\widetilde{\mathbf{x}}, y|\mathbf{X})$ can be implemented by the following pseudo code.

Algorithm 1 Data Sampling

```
infer p(U, V, z|X_−) and p(U, V, z|X_+) using SIBM;
sample V_+, z_+ from p(U, V, z|X_+);
sample V_−, z_− from p(U, V, z|X_−);
for each new instance (x̃, y) do
    randomly sample y from {+, −};
    randomly sample i from {1, ..., L_y};
    sample u_i from p(U, V, z|X_y);
    for j in 1...M do
        sample x̃_j from Ber(x̃_j|δ(u_i · v_{j,y} + z_y));
    end for
end for
```

This algorithm allows us to sample infinite instances from the fixed distribution $p(\widetilde{\mathbf{x}}, y|\mathbf{X})$, to be used for generating data of various sizes. Next, we will discuss how to systematically vary sparseness of the sampled data.

2.4 Data Missing Model

We employ a simple data missing model to add and vary data sparseness. Our data missing model assumes that each attribute has the same probability to become 0, which follows the Missing Completely At Random assumption [Heitjan and Basu, 1996].

Given the probability of missing s, the data missing model will transform an instance $\widetilde{\mathbf{x}} = (\widetilde{x}_1, \widetilde{x}_2, ..., \widetilde{x}_m)$ to $\mathbf{x} = (x_1, x_2, ..., x_m)$ following:

$$p(\mathbf{x}|\widetilde{\mathbf{x}}, s) = \prod_{j=1}^{m} Ber(\widetilde{x}_j \cdot (1 - s)) \quad (10)$$

To ease our illustration, we hereafter call this process as *dilution*, and the resultant data as the *diluted data*. Now if instances are originally sampled from $p(\widetilde{\mathbf{x}}|y)$, the distribution of the diluted data can be computed by $p(\mathbf{x}|y, s) = \int p(\mathbf{x}|\widetilde{\mathbf{x}}, s) p(\widetilde{\mathbf{x}}|y) d\widetilde{\mathbf{x}}$. When we apply this dilution process to data generated from Algorithm 1, we denote the resultant distribution as $p(\mathbf{x}|y, \mathbf{X}, s) = \int p(\mathbf{x}|\widetilde{\mathbf{x}}, s) p(\widetilde{\mathbf{x}}|y, \mathbf{X}) d\widetilde{\mathbf{x}}$.

Using the above data missing model, additional sparseness will be introduced uniformly to each attribute and higher s will lead to sparser data. This enables us to vary s and study the impact of data sparseness systematically, as we will describe in the next section.

3 Experiment

In this section, we describe how we use the proposed data generation and dilution process to study our research question with experiments.

Data. We use two movie rating datasets: the movielens 1M dataset[1] (*ml-1m*) and the Yahoo Movies dataset[2] (*ymovie*). As mentioned earlier, we only consider absence/presence of each rating. The preprocessed *ml-1m* dataset is $3,418$ (instances) $\times 3,647$ (attributes) with balanced classes and an original sparseness of 0.9550. The preprocessed *ymovie* dataset is $9,489$ (instances) $\times 4,368$ (attributes) with balanced classes and an original sparseness of 0.9977.

Training. To study the impact of data sparseness, we use various values of s; see the legends of Figure 1 and 2. For each s, we generate training samples of various sizes l from $p(\mathbf{x}, y|\mathbf{X}, s)$ using the proposed data generation and dilution process. We use Lib-linear [Fan *et al.*, 2008] with default parameters to train the classifier and get the corresponding training error $\epsilon_{train}(l, s)$.

Testing. It is empirically impossible to test a classifier on the whole distribution and get the true generalization error. For this reason, we generate a large test dataset from $p(\mathbf{x}, y|\mathbf{X}, s)$ for each setting of s. For the *ymovie* experiment, the size of test sets is 7.59 million instances (800 times of the real data size). For the *ml-1m* experiment, the test size is 0.68 million (200 times of the real data size). We test the classifiers on the corresponding test set and get the test error $\epsilon_{test}(s)$.

For each setting of s and training size l, we repeat the data generation (including dilution), training and testing process for 10 times, and record the average training and testing error rates. The most time consuming step in our experiment is data generation: the largest training dataset ($l = 2^{20}$ in the *ymovie* experiment) costs us one day on a modern Xeon CPU to generate each time; Meanwhile, training on this particular dataset only costs several minutes, thanks for the efficiency of linear SVM. To speed up, we straight-forwardly parallelize the data generation jobs on a cluster.

As we systematically vary s, the resultant learning curves of averaged training and testing error rates are given in Figure 1. Figure 2 shows the difference between the two errors in order to show the rate of convergence. We have two important observations which are obvious in both experiments:

[1] http://grouplens.org/datasets/movielens/

[2] http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

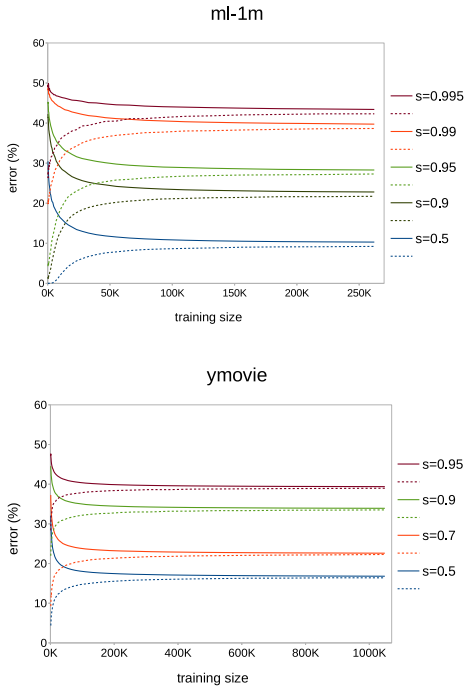

Figure 1: Training (dashed) and generalization error rates for different data missing probability s. **Observation:** higher sparseness leads to larger asymptotic generalization error rate.

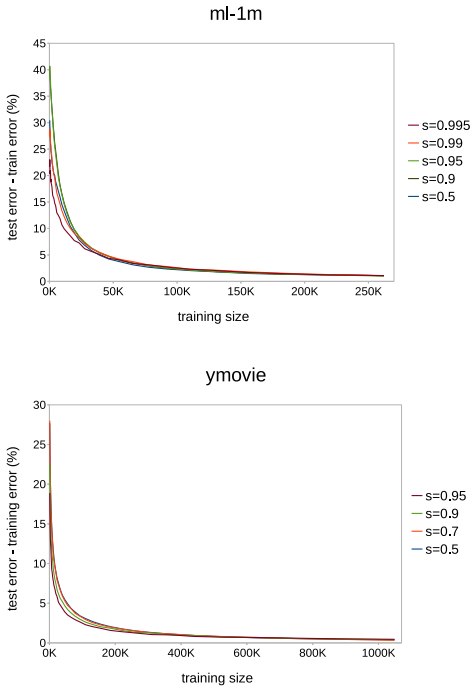

Figure 2: The difference between training and generalization error rates for different data missing probability s. **Observation:** asymptotic rate of convergence is almost the same for different sparseness.

Observation 1. *asymptotic generalization error*: Higher sparseness leads to larger asymptotic generalization error rate;

For example, in the *ymovie* experiment, the asymptotic (training size $l > 10^6$) generalization error rates are 16.6%, 22.4%, 33.7% and 39.2% for $s = 0.5, 0.7, 0.9$ and 0.95, respectively.

Observation 2. *asymptotic rate of convergence:* the asymptotic rate of convergence is almost the same for different sparseness.

In other words, different sparseness would not change the amount of training data needed to approach convergence. For example, in the *ymovie* experiment, the minimum training size l for the two error rates to be within 1% is almost the same ($l = 370K$), regardless of the value of s.

In the next section, we will study the theoretic reasons behind these observations.

4 Theoretic Analysis

In section 4.1, we study the theoretic reason for observation 1, and in section 4.2 we show that the theoretic reason of observation 2 can be easily explained using PAC bound.

4.1 Asymptotic Generalization Error

Suppose: (1) the original data distribution is $p(\mathbf{x}|y)$;[3] (2) after applying the data missing model described in Section 2.4, the diluted data distribution is $p(\mathbf{x}|y, s)$; (3) class prior is balanced, i.e., $p(+) = p(-) = 0.5$. From the Bayesian Decision Theory [Duda *et al.*, 2012], we know that the asymptotic generalization error rate of linear SVM can be lower bounded by the Bayes risk, which is the best classification performance that any classifier can achieve over a given data space:

$$R = \int \min_{y \in \{-,+\}} \{R(y(\mathbf{x})|\mathbf{x})\} p(\mathbf{x}) d\mathbf{x} \tag{11}$$

where $R(y(\mathbf{x})|\mathbf{x})$ is the loss for predicting the label of instance $\mathbf{x}$ as y, which in our case is the 0-1 loss. We denote the Bayes risk for $p(\mathbf{x}|y, s)$ as $R(s)$, thus the asymptotic generalization error rate is lower bounded:

$$\lim_{l \to \infty} \epsilon(s) \geq R(s) \tag{12}$$

Notice $\mathbf{x}$ lives in the discrete data space $\{0, 1\}^m$, which eases us to write the Bayes risk as the following form:

$$R(s) = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \min_{y \in \{+,-\}} p(y|\mathbf{x}, s)$$
$$= \sum_{\mathbf{x}} \min_{y \in \{+,-\}} p(y, \mathbf{x}|s) = 0.5 \sum_{\mathbf{x}} \min_{y \in \{+,-\}} p(\mathbf{x}|y, s) \tag{13}$$

[3] Our proof does not require $p(\mathbf{x}|y)$ to be the specific distribution $p(\widetilde{\mathbf{x}}|y, \mathbf{X})$ introduced earlier.

We will next prove that higher s leads to larger $R(s)$ using three steps. We first consider the case if only one of the attributes is diluted. In this case, we could prove that the Bayes risk will not decrease by any chance, and with high probability it will increase (Lemma 4.1). We next consider the case if we still dilute only one attribute but with different s, we prove that higher sparseness will lead to larger Bayes risk (Lemma 4.2). Based on these results, we finally prove that higher s leads to larger Bayes risk $R(s)$ (Theorem 4.3).

When we only dilute one of the m attributes, x_j, we denote the rest of the attributes as $\mathbf{x}_{-j} = (x_1, ...x_{j-1}, x_{j+1}, ..., x_m)$. Since the order of attributes does not matter, we denote $\mathbf{x}$ as $(\mathbf{x}_{-j}, x_j)$. We denote the corresponding distribution as $p^{(j)}(\mathbf{x}|y, s)$ and Bayes risk as $R^{(j)}(s)$, we now prove:

Lemma 4.1 *$R^{(j)}(s) \geq R(0)$ always holds. Specifically, with high probability, $R^{(j)}(s) > R(0)$.*

Proof Since we have only diluted x_j, we first expand the computation of Bayes risk (13) along x_j:

$$R^{(j)}(s) - R(0) = 0.5 \sum_{\mathbf{x}_{-j}} [Z^{(j)}(\mathbf{x}_{-j}, s) - Z(\mathbf{x}_{-j})] \quad (14)$$

where $Z(\mathbf{x}_{-j})$ denotes the sum of probability mass of $p(\mathbf{x}|y)$ at $(\mathbf{x}_{-j}, 0)$ and $(\mathbf{x}_{-j}, 1)$, each minimized among classes:

$$Z(\mathbf{x}_{-j}) := \min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 0|y) + \min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 1|y) \quad (15)$$

$Z^{(j)}(\mathbf{x}_{-j}, s)$ is also defined accordingly for $p^{(j)}(\mathbf{x}|y, s)$. $\sum_{\mathbf{x}_{-j}}$ denotes the summation over all $\mathbf{x}_{-j}$ in the $\{0, 1\}^{m-1}$ space. Now define $\Delta_Z(s) := Z^{(j)}(\mathbf{x}_{-j}, s) - Z(\mathbf{x}_{-j})$, we will next prove that $\Delta_Z(s) \geq 0$ holds for all $\mathbf{x}_{-j} \in \{0, 1\}^{m-1}$:

Since x_j is diluted with s, according to (10), this means $\forall \mathbf{x}_{-j} \in \{0, 1\}^{m-1}$, we have

$$p^{(j)}(\mathbf{x}_{-j}, 0|y, s) = p(\mathbf{x}_{-j}, 0|y) + s \cdot p(\mathbf{x}_{-j}, 1|y) \quad (16)$$

$$p^{(j)}(\mathbf{x}_{-j}, 1|y, s) = (1 - s) \cdot p(\mathbf{x}_{-j}, 1|y) \quad (17)$$

We denote $y_h \in \{-, +\}$ as the label that has a higher probability mass at $(\mathbf{x}_{-j}, 1)$, and we denote the other label as y_l:[4]

$$p(\mathbf{x}_{-j}, 1|y_h) \geq p(\mathbf{x}_{-j}, 1|y_l), \quad (18)$$

$$\min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 1|y) = p(\mathbf{x}_{-j}, 1|y_l) \quad (19)$$

and (17)(18) lead to:

$$\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 1|y, s) = p^{(j)}(\mathbf{x}_{-j}, 1|y_l, s) \quad (20)$$

In order to write out $\min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 0|y)$ and $\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 0|y, s)$, we define

$$g(\mathbf{x}_{-j}) := \frac{p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h)}{p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)} \quad (21)$$

[4]Notice that y_h and y_l will change for different $\mathbf{x}_{-j}$. Strictly, we should use the notation $y_h(\mathbf{x}_{-j}, 1)$ and $y_l(\mathbf{x}_{-j}, 1)$ if not for the purpose of brevity.

For each $\mathbf{x}_{-j}$ there are in total three different situations to consider:

$$\textbf{Case 1}. \quad p(\mathbf{x}_{-j}, 0|y_h) \geq p(\mathbf{x}_{-j}, 0|y_l) \quad (22)$$

$$\textbf{Case 2}. \quad \begin{cases} p(\mathbf{x}_{-j}, 0|y_h) < p(\mathbf{x}_{-j}, 0|y_l) \\ s \geq g(\mathbf{x}_{-j}) \end{cases} \quad (23)$$

$$\textbf{Case 3}. \quad \begin{cases} p(\mathbf{x}_{-j}, 0|y_h) < p(\mathbf{x}_{-j}, 0|y_l) \\ s < g(\mathbf{x}_{-j}) \end{cases} \quad (24)$$

We could straight-forwardly compute $\Delta_Z(s)$ for each case. Case 1:

$$\Delta_Z(s) = 0 \quad (25)$$

Case 2:

$$\Delta_Z(s) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h) > 0 \quad (26)$$

Case 3:

$$\Delta_Z(s) = s \cdot [p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)] > 0 \quad (27)$$

For space limit, detailed derivation of (25)(26)(27) is omitted.

Now that $\Delta_Z(s) \geq 0$ always holds, $R^{(j)}(s) \geq R(0)$ is true because of (14). Moreover, $R^{(j)}(s) = R(0)$ is true only if Case 1 happens for all $\mathbf{x}_{-j} \in \{0, 1\}^{m-1}$, which has low probability. ∎

In the next lemma, we consider the case when we vary the sparseness s on the one diluted attribute.

Lemma 4.2 *$\forall s_1, s_2$ s.t. $1 \geq s_2 > s_1 \geq 0$, we have $R^{(j)}(s_2) \geq R^{(j)}(s_1)$. Specifically:*

(1). only when both s_1 and s_2 are close enough to 1.0, $R^{(j)}(s_2) = R^{(j)}(s_1)$ will be true with high probability.

(2). other wise, $R^{(j)}(s_2) > R^{(j)}(s_1)$.

Its proof could be derived by studying the value of $\Delta_Z(s)$ in different situations. It is straight-forward after we have derived equations (25)(26)(27) in the proof of Lemma 4.1.

Proof We first prove that $\Delta_Z(s_2) \geq \Delta_Z(s_1)$ always holds. For any given $\mathbf{x}_{-j} \in \{0, 1\}^{m-1}$:

(1). If Case 1 is true, we will always get $\Delta_Z(s_2) = \Delta_Z(s_1) = 0$ regardless of the value of s_1 and s_2.

(2). If Case 1 is false and $1 \geq s_2 > s_1 \geq g(\mathbf{x}_{-j})$, then Case 2 holds for both s_1 and s_2, and we get

$$\Delta_Z(s_1) = \Delta_Z(s_2) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h)$$

(3). If Case 1 is false and $s_2 \geq g(\mathbf{x}_{-j}) > s_1$, then Case 2 holds for s_2 and Case 3 holds for s_1. Thus

$$\Delta_Z(s_2) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h) > s_1 \cdot [p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)] = \Delta_Z(s_1)$$

(4). Finally, if Case 1 is false and $g(\mathbf{x}_{-j}) > s_2 > s_1 \geq 0$, then Case 3 holds for both s_1 and s_2. From (27) we know that $\Delta_Z(s_2) > \Delta_Z(s_1)$.

Now we have proved that $\Delta_Z(s_2) \geq \Delta_Z(s_1)$ always hold, substitute into (14) leads to

$$R^{(j)}(s_2) - R^{(j)}(s_1) = [R^{(j)}(s_2) - R(0)] - [R^{(j)}(s_1) - R(0)] = \sum_{\mathbf{x}_{-j}} [\Delta_Z(s_2) - \Delta_Z(s_1)] \geq 0 \quad (28)$$

From the above analysis, we further notice that $R^{(j)}(s_2) = R^{(j)}(s_1)$ only happens if for $\forall \mathbf{x}_{-j} \in \{0,1\}^{m-1}$ either Case 1 happens or Case 2 holds for both s_1 and s_2. Notice for given $p(\mathbf{x}, y)$, it is very unlikely that all $\mathbf{x}_{-j}$ could satisfy Case 1. For those $\mathbf{x}_{-j}$ that Case 1 is not satisfied, it thus requires Case 2 to hold for both s_1 and s_2, which is only likely when both s_1 and s_2 are close enough to 1.0. ∎

The next theorem will generalize to the case when sparseness is introduced to all attributes, which is the goal of our proof:

Theorem 4.3 *$\forall s_1, s_2$ s.t. $1 \geq s_2 > s_1 \geq 0$, we have $R(s_2) \geq R(s_1)$. Specifically:*

(1). only when both s_1 and s_2 are close enough to 1.0, $R(s_2) = R(s_1)$ will be true with high probability.

(2). other wise, $R(s_2) > R(s_1)$.

The basic idea of our proof is to find a hypothetic process that varies the dilution of data from s_2 to s_1 one attribute at a time, so we could leverage the result of lemma 4.2.

Proof We use F to denote the total attribute set $\{x_1, ..., x_m\}$. At a certain state, we use $F(s_2) \subset F$ to denote the set of attributes that have been diluted by s_2; and $F(s_1) \subset F$ the set of attributes that have been diluted by s_1. We denote the corresponding distribution as $p^{F(s_1),F(s_2)}(\mathbf{x}|y, s_1, s_2)$ and its Bayes Risk as $R^{F(s_1),F(s_2)}(s_1, s_2)$.

Now we consider the following process that iteratively changes the elements of $F(s_1)$ and $F(s_2)$: we start from $F(s_1) = \emptyset \wedge F(s_2) = F$, and move each attribute in $F(s_2)$ to $F(s_1)$ one after another. After m such steps we come to $F(s_1) = F \wedge F(s_2) = \emptyset$.

After step i, we denote the current $F(s_1)$ as $F_i(s_1)$, $F(s_2)$ as $F_i(s_2)$. Using this notation, the initial state is $F_0(s_1) = F \wedge F_0(s_2) = \emptyset$, and final state is $F_m(s_1) = \emptyset \wedge F_m(s_1) = F$. It's obvious that

$$R^{F_0(s_1),F_0(s_2)}(s_1, s_2) = R(s_1)$$

$$R^{F_m(s_1),F_m(s_1)}(s_1, s_2) = R(s_2)$$

We next prove that for $\forall i \in \{0, ..., m-1\}$:

$$R^{F_{i+1}(s_1),F_{i+1}(s_2)}(s_1, s_2) \geq R^{F_i(s_1),F_i(s_2)}(s_1, s_2) \quad (29)$$

Notice that in each step i, there is only one attribute (e.g., x_j) changes from $F_i(s_2)$ to $F_i(s_1)$, i.e., $F - [F_i(s_1) \cup F_{i+1}(s_2)] = \{x_j\}$.

Now consider the distribution $p^{F_i(s_1),F_{i+1}(s_2)}(\mathbf{x}|y, s_1, s_2)$, which corresponds to an intermediate state between step i and $i+1$ with x_j being not diluted. Now if we dilute x_j by s_2, we get $p^{F_{i+1}(s_1),F_{i+1}(s_2)}(\mathbf{x}|y, s_1, s_2)$; Instead, if we dilute x_j by s_1, we get $p^{F_i(s_1),F_i(s_2)}(\mathbf{x}|y, s_1, s_2)$. Using Lemma 4.2, it now becomes obvious that (29) is true, which further leads to

$$R(s_2) = R^{F_m(s_1),F_m(s_1)}(s_1, s_2)$$

$$\geq ... \geq R^{F_0(s_1),F_0(s_2)}(s_1, s_2) = R(s_1) \quad (30)$$

We further notice that $R(s_2) = R(s_1)$ only holds when all m equal signs hold simultaneously. As illustrated in Lemma 4.2, each equal sign holds with high probability only when both s_2 and s_1 are close enough to 1.0. ∎

This theorem tells us that higher sparseness leads to larger Bayes risk, which explains our observation on the asymptotic generalization error.

This result is important for understanding how data sparseness affects the asymptotic generalization error of linear SVM.

Since the Bayes risk of the data space is irrelevant to the classifier in use, our theoretic result is also applicable for other classifiers.

4.2 Asymptotic Rate of Convergence

Our observation on the asymptotic rate of convergence could be explained by the PAC bound [Bartlett and Shawe-Taylor, 1999]. From the PAC bound, we know that the convergence of generalization error rate ϵ and the training error rate ϵ_{tr} is bounded in probability by training size l and the VC-dimension $|d|$:

$$Pr\left\{\epsilon > \epsilon_{tr} + \sqrt{\frac{\ln|d| - \ln\delta}{2l}}\right\} \leq \delta \quad (31)$$

For fixed δ, the rate of convergence is approximately:

$$\frac{\partial(\epsilon - \epsilon_{tr})}{\partial l} \approx -\sqrt{\frac{\ln\frac{|d|}{\delta}}{2l^3}} \quad (32)$$

Though different data sparseness could change $|d|$, for linear SVM, $|d| < m + 1$ holds true regardless of data sparseness.[5] From (32) we could see that asymptotically (when $2l^3 \gg \ln\frac{m+1}{\delta}$), the rate of convergence is mostly influenced by the increase of training size l rather than by the change of $|d|$, since $|d|$ is always upper bounded by $m + 1$. In other words, varying sparseness will have little impact on the asymptotic rate of convergence, which verifies our observation.

When using linear SVM in real-world applications, it is important to know whether sparser data would lead to slower convergence rate. If so, practitioners will have to collect more training instances in order for linear SVM to converge on highly sparse datasets. Our observation and analysis shows that the rate of convergence is almost not affected by different data sparseness for linear SVM.

5 Conclusion

Linear SVM is efficient for classifying large sparse datasets. In order to understand how data sparseness affects the convergence behavior of linear SVM, we propose a novel approach to generate large and sparse data from real-world datasets using statistical inference and the data sampling process of the PAC framework. From our systematic experiments, we have observed: 1. Higher sparseness will lead to larger asymptotic generalization error rate; 2. Convergence rate of learning is almost unchanged for different sparseness. We have also proved these findings theoretically. Our experiment and theoretic results are valuable for learning large sparse datasets with linear SVM.

[5] Notice that VC-dimension can be very large for some non-linear versions of SVM, how their convergence rate will be affected by data sparseness could be an interesting future work.

Acknowledgments

This work is supported by National Natural Science Foundation of China (grants 61432020, 61472430, 61202137) and Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[Bartlett and Shawe-Taylor, 1999] Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. *Advances in Kernel MethodsSupport Vector Learning*, pages 43–54, 1999.

[Bennett and Lanning, 2007] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[Cha *et al.*, 2009] Meeyoung Cha, Alan Mislove, and Krishna P Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World wide web*, pages 721–730. ACM, 2009.

[Chechik *et al.*, 2008] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel, and Daphne Koller. Max-margin classification of data with absent features. *The Journal of Machine Learning Research*, 9:1–21, 2008.

[Duda *et al.*, 2012] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Heitjan and Basu, 1996] Daniel F Heitjan and Srabashi Basu. Distinguishing missing at random and missing completely at random. *The American Statistician*, 50(3):207–213, 1996.

[Lobato *et al.*, 2014] Jóse Miguel Hernández Lobato, Neil Houlsby, and Zoubin Ghahramani. Stochastic inference for scalable probabilistic modeling of binary matrices. In *Proceedings of The 31st International Conference on Machine Learning*, page to appear, 2014.

[Long and Servedio, 2013] Philip M Long and Rocco A Servedio. Low-weight halfspaces for sparse boolean vectors. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 21–36. ACM, 2013.

[Mnih and Salakhutdinov, 2007] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[Pelckmans *et al.*, 2005] Kristiaan Pelckmans, Jos De Brabanter, Johan AK Suykens, and Bart De Moor. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5):684–692, 2005.

[Shalev-Shwartz *et al.*, 2011] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[Vapnik, 1999] Vladimir N Vapnik. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5):988–999, 1999.