Enhancing SVMs with Problem Context Aware Pipeline

Zeyi Wen[†], Zhishang Zhou[§], Hanfeng Liu[‡], Bingsheng He[‡], Xia Li[♯], Jian Chen[§]
zeyi.wen@uwa.edu.au,{leonzhou6171,kurt.liuhf}@gmail.com,dcsheb@nus.edu.sg,{xiali@mail.gdufs,ellachen@scut}.edu.cn
[†]The University of Western Australia, [‡]National University of Singapore, [§]South China University of Technology,
[♯]Guangdong University of Foreign Studies, China

ABSTRACT

In recent years, many data mining practitioners have treated deep neural networks (DNNs) as a standard recipe of creating the state-of-the-art solutions. As a result, models like Support Vector Machines (SVMs) have been overlooked. While the results from DNNs are encouraging, DNNs also come with their huge number of parameters in the model and overheads in long training/inference time. SVMs have excellent properties such as convexity, good generality and efficiency. In this paper, we propose techniques to enhance SVMs with an automatic pipeline which exploits the context of the learning problem. The pipeline consists of several components including data aware subproblem construction, feature customization, data balancing among subproblems with augmentation, and kernel hyper-parameter tuner. Comprehensive experiments show that our proposed solution is more efficient, while producing better results than the other SVM based approaches. Additionally, we conduct a case study of our proposed solution on a popular sentiment analysis problem—the aspect term sentiment analysis (ATSA) task. The study shows that our SVM based solution can achieve competitive predictive accuracy to DNN (and even majority of the BERT) based approaches. Furthermore, our solution is about 40 times faster in inference and has 100 times fewer parameters than the models using BERT. Our findings can encourage more research work on conventional machine learning techniques which may be a good alternative for smaller model size and faster training/inference.

CCS CONCEPTS

• **Computing methodologies → Supervised learning by classification**.

KEYWORDS

Support Vector Machines, Machine Learning, Sentiment Analysis

ACM Reference Format:
Zeyi Wen[†], Zhishang Zhou[§], Hanfeng Liu[‡], Bingsheng He[‡], Xia Li[♯], Jian Chen[§]. 2021. Enhancing SVMs with Problem Context Aware Pipeline. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467291

1 INTRODUCTION

Deep neural networks (DNNs) have achieved great success in many areas including image processing and text mining. While the popularity of DNNs and the promising results are encouraging, DNNs also come with their huge number of parameters in the model and overheads in training/inference time. In comparison, Support Vector Machines (SVMs) have excellent properties such as convexity, good generality and efficiency, but SVMs have been overlooked in recent years. SVMs may be a model to satisfy applications with stricter requirements of model sizes and training/inference time.

In this paper, we develop techniques to enhance the performance of SVMs by exploiting a pipeline which uses the context of the learning problem. Unlike neural networks where the number of learnable parameters can be easily adjusted (e.g., adding more layers or neurons) for different problems, the number of learnable parameters for SVMs cannot be easily increased. For example, the dimension of the weight vector in the prime form of SVMs, and the number of Lagrange multipliers in the dual form of SVMs cannot be changed for a problem [29]. Our proposed solution demonstrates that equipping SVMs with an automatic pipeline to exploit the context of a learning problem can improve performance. The pipeline optimizes the important operations such as subproblem construction, feature customization, augmentation for data balancing, and tuning SVM kernel hyper-parameters. Moreover, the pipeline is powered by forward and backward propagation techniques to guide the SVM training and other important operations within the pipeline.

The training of SVMs with problem context aware pipeline works as follows. First, the learning problem is divided into multiple subproblems based on similarity among the training instances, and then an SVM classifier is dedicated to each subproblem. As a result, each SVM classifier can be specifically optimized to better fit the subproblem. Second, the automatic feature customization is performed for each base SVM classifier during the training. As the class distribution in the subproblems tends to be unbalanced, we augment the data by exploiting data of other subproblems. Third, a hyper-parameter tuner is dedicated to optimize the hyper-parameters of SVMs, so that the kernel and the regularization hyper-parameters are automatically tuned for each SVM classifier. Finally, the error/loss produced by the SVM classifiers is backward propagated to further optimize the pipeline until the termination condition is met. We formulate the learning problem of SVMs with the problem context aware pipeline, and theoretically prove that the SVMs with the pipeline are better than the single SVM based approach.

We conduct experiments to evaluate our proposed solution in comparison with existing SVMs on several data sets. The results show that our solution is more efficient, while producing better results than the other SVM based approaches. Moreover, we perform a case study on a popular customer review analysis problem—the aspect term sentiment analysis (ATSA) task. The ATSA task [11] aims to identify the polarity (e.g., positive, negative, neutral) of each aspect (e.g., food and service) rather than the polarity of the whole review. The finer granularity of analysis in ATSA brings new challenges in building the classifier for sentiment analysis, and the traditional SVM based approaches are unlikely to be promising [11]. We demonstrate that our SVM based approach can achieve competitive predictive accuracy to DNN based approaches, and even outperforms majority of the BERT based approaches in the ATSA task. Our theoretical analysis also shows that the training time complexity of our solution is lower than the DNN based approaches. Hence, our solution can train models efficiently on multi-core CPUs, and can be much faster using GPUs. In comparison, the DNN based methods heavily rely on special hardware such as GPUs and TPUs. To summarize, we make the following major contributions in this paper.

- We formulate the learning problem of SVMs with problem context aware pipeline. We prove that our solution is theoretically better than the single SVM based approach, and can automatically consider the context of a learning problem for achieving better predictive accuracy.
- We propose a series of techniques to power the problem context aware pipeline, including data aware subproblem construction, feature customization for each subproblem, data augmentation to tackle data unbalancing among the subproblems, and automatic kernel and regularization parameter tuning.
- We experimentally demonstrate that our proposed solution is more efficient, while producing better results than the other SVM based approaches. Our case study on the ATSA task shows that our SVM based solution can achieve competitive predictive accuracy to DNN (and even BERT) based approaches. Our solution is fast to train due to much lower time complexity. The efficiency on inference is about 40 times faster and the trained model has 100 times fewer parameters than the models using BERT.

2 OUR PROPOSED SOLUTION

In this section, we present the details of our proposed SVM solution equipped with a problem context aware pipeline. The key idea of our solution is to incorporate more information from a learning problem into the training process. First, our proposed solution divides the learning problem into multiple subproblems based on similarity of the training instances, such that a subproblem can be well addressed by an SVM classifier specifically optimized for it. This process trains more SVM classifiers to perform finer granularity optimization, instead of using only one SVM classifier as many of the previous SVM based approaches do [11, 28]. Second, we enable the training process to automatically tune the kernel and regularization hyper-parameters (e.g., kernel type and the kernel hyper-parameters), rather than manually setting them. We theoretically prove that our proposed solution is better than the one SVM based approaches. However, the SVMs with the problem context aware pipeline comes with previously unseen challenges, including the need of tackling data unbalancing issues among subproblems,

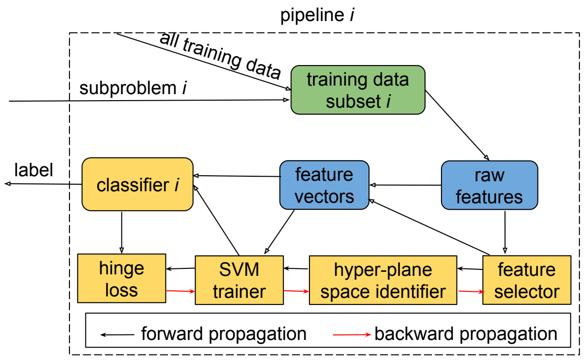


Figure 1: The pipeline of SVM training for a subproblem

and feature customization for each individual SVM. To tackle those challenges, we propose a series of novel techniques to ensure the model quality. We elaborate the whole training process in greater details in the rest of this section.

Overview of our solution: Figure 1 gives a high-level overview of our proposed pipeline. First, we construct subproblems by clustering to group the training data into non-overlapping subsets, so that similar training instances are grouped together. For each subproblem, an SVM classifier is trained and optimized on the corresponding subset of training data. Specifically, we perform feature selection on the raw features to build the customized set of features for the subproblem. Then, the selected features form a feature vector for each training instance (cf. Figure 1 center). Meanwhile, we have a hyper-plane space identifier for setting proper hyper-parameters for SVMs (i.e., identifying a space for the separating hyper-plane). After that, the feature vectors and the hyper-parameters together are fed to the SVM trainer to learn an SVM classifier. The loss/error of the classifier is computed and backpropogated to improve the feature selector, hyper-plane space identifier and SVM trainer. Therefore, the SVM classifier trained in this process is well-tuned and customized for the subproblem, as the features, hyper-parameters and the parameters of the SVM classifier are thoroughly learned automatically. Finally, those SVM classifiers together form the final classifier for the problem.

2.1 Problem Formulation

We formulate SVMs with problem context aware pipeline as the "SVM-CAP" learning problem here. Let $\mathcal{T}$ and $\mathcal{V}$ denote the training data set and the validation data set, respectively. The training set and validation set are further clustered into k subsets denoted by $\mathcal{T}^1, \ldots, \mathcal{T}^k$ and $\mathcal{V}^1, \ldots, \mathcal{V}^k$, respectively. The learnable hyper-parameters include: $\mathcal{F}$ which is a set of features; $\mathcal{H}$ which is the candidate SVM kernel types; Λ which contains the kernel hyper-parameters and the regularization constant of SVMs; and k which is the number of subproblems/subsets. Thus, the learnable hyper-parameters space can be defined as $\Theta = \mathcal{F} \times \mathcal{H} \times \Lambda$. Then, the learning problem is to minimize the following objective function:

$$\underset{k \in \mathbb{N}^+, \theta^i \in \Theta}{\arg\min} \sum_{i=1}^{k} \frac{|\mathcal{V}^i|}{|\mathcal{V}|} \mathcal{L}(\theta^i, \mathcal{T}^i, \mathcal{V}^i)$$

where $\mathcal{L}(\theta^i, \mathcal{T}^i, \mathcal{V}^i)$ denotes the loss of the i-th SVM on $\mathcal{V}^i$, $\theta^i = \{f^i, h^i, \lambda^i\}$, f^i denotes the features used in the i-th SVM, h^i is the used SVM kernel type, λ^i denotes the corresponding kernel hyper-parameters, $\frac{|\mathcal{V}^i|}{|\mathcal{V}|}$ is the weight of the i-th SVM and $\mathbb{N}^+$ is the set of positive natural numbers.

Optimization on the SVM-CAP problem: The idea of the gradient based approaches can be used to solve the SVM-CAP problem. More specifically, the derivative of the learning problem over θ^i is $\sum_{i=1}^{k} \frac{|\mathcal{V}^i|}{|\mathcal{V}|} \cdot \frac{\partial \mathcal{L}(\theta^i, \mathcal{T}^i, \mathcal{V}^i)}{\partial \theta^i}$. Since $\theta^i = \{f^i, h^i, \lambda^i\}$, the derivative can be written as $\sum_{i=1}^{k} \frac{|\mathcal{V}^i|}{|\mathcal{V}|} \cdot (\frac{\partial \mathcal{L}(\theta^i, \mathcal{T}^i, \mathcal{V}^i)}{\partial f^i \cdot h^i \cdot \lambda^i} + \frac{\partial \mathcal{L}(\theta^i, \mathcal{T}^i, \mathcal{V}^i)}{\partial h^i \cdot f^i \cdot \lambda^i} + \frac{\partial \mathcal{L}(\theta^i, \mathcal{T}^i, \mathcal{V}^i)}{\partial \lambda^i \cdot f^i \cdot h^i})$. As $\Theta = \mathcal{F} \times \mathcal{H} \times \Lambda$ has discrete and conditional variables (e.g., the degree d in Λ is discrete and is used only for the polynomial kernel), there is no closed form for computing the gradients. In our solution, we use the sequential model-based optimization with Tree Parzen Estimator and Expected Improvement [2] to solve the SVM-CAP problem. This method can solve optimization problems where the search space is noncontinuous and conditional variables exist.

2.1.1 The Generalization Bound of Our Proposed Solution. Fundamentally, our solution tackles a learning problem with multiple SVMs, while the mainstream SVM based solutions use one SVM [28]. Here we theoretically demonstrate that our solution leads to a better generalization bound.

Theorem 2.1 (Margin bound for multi-class classification with multiple multi-class SVMs). *Let $\mathcal{X}$ denotes the input space and $\mathcal{Y} = \{1, 2, \ldots, k\}$ denotes the output space, where $k > 2$. Let $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ be a positive definite symmetric (PDS) kernel and $\Phi : \mathcal{X} \rightarrow \mathbb{H}$ be a feature mapping associated to K. Assume that there exists $r > 0$ such that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$. We define the piecewise kernel-based hypothesis space as $\bar{\mathcal{H}}_{K,p} = \{(x, y) \in \mathcal{X} \times \mathcal{Y} \rightarrow \bar{w}_y \cdot \Phi(x) : \bar{W} = (\bar{w}_1, \ldots, \bar{w}_k)^\top, \bar{w}_l = ConditionalOptimal(\bar{w}_{1,l}, \ldots, \bar{w}_{c,l})$ for all $l \in \{1, 2, \ldots, k\}$ where $ConditionalOptimal()$ represents a piecewise function defined on a sequence of intervals consisting of $x \in \mathcal{X}$ satisfying specific condition and c denotes the number of pieces, $||\bar{W}||_{\mathbb{H},p} = (\sum_{l=1}^{k} ||\bar{w}_l||_{\mathbb{H}}^{p})^{1/p} \leq (\sum_{l=1}^{k} ||\bar{w}_{*,l}||_{\mathbb{H}}^{p})^{1/p} \leq \bar{\Lambda}$ for any $p \geq 1$ where $||\bar{w}||_{\mathbb{H}} = \sqrt{\bar{w}^T \bar{w}}$ and $||\bar{w}_{*,l}||_{\mathbb{H}} = max\{||\bar{w}_{1,l}||_{\mathbb{H}}, \ldots, ||\bar{w}_{c,l}||_{\mathbb{H}}\}$. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following multi-class classification generalization bounds of multiple multi-class SVM holds for all $h \in \bar{\mathcal{H}}_{K,p}$.*

$$R(h) \leq \frac{1}{m} \sum_{i=1}^{m} \bar{\xi}_i + 4k\sqrt{\frac{r^2\bar{\Lambda}^2}{m}} + \sqrt{\frac{log\frac{1}{\delta}}{2m}}, \quad (1)$$

where $\bar{\xi}_i = max\{1 - [\bar{w}_{y_i} \cdot \Phi(x_i) - max_{y' \neq y_i} \bar{w}_{y'} \cdot \Phi(x_i)], 0\}$ for all $i \in \{1, 2, \ldots, m\}$.

The proof of the above theorem is provided in the Appendix. Compared with existing generalization bound of single SVM $R(h) \leq \frac{1}{m} \sum_{i=1}^{m} \xi_i + 4k\sqrt{\frac{r^2\Lambda^2}{m}} + \sqrt{\frac{log\frac{1}{\delta}}{2m}}$ [18], our bound shown in (1) is tighter since $\bar{\xi} \leq \xi$ and $\bar{\Lambda} \leq \Lambda$. This result shows that SVMs with the problem context aware pipeline can improve the model quality, which is particularly true for our case study on the ATSA task where a single SVM classifier is insufficient to fit the whole sentiment analysis problem.

2.2 Subproblem Construction, Data Augmentation and Feature Customization

Here, we elaborate the key components of the problem context aware pipeline, which aims to make efficient use of more information of a problem in the model building process.

2.2.1 Subproblem construction. As we have discussed earlier in this section, using a single SVM classifier to deal with a complex problem may lead to poor predictive accuracy, due to the limited model capacity of one SVM classifier. In our proposed solution, we first divide a problem into subproblems, where each subproblem contains training instances sharing similar information (e.g., similar semantics for text mining problems like the ATSA task). Hence, our solution is able to use more SVMs to handle complex problems, and each SVM classifier is specifically trained for a subproblem.

The subproblem construction is important for complex problems such as the ATSA task. The key intuition is that an aspect may be described using different aspect terms. For instance, aspect terms including "manager", "staff" and "chef" may be used to rate the "personnel" aspect of a restaurant. Moreover, similar aspect terms tend to be described by similar adjectives. For example, adjectives including "delicious" and "yummy" may be used to describe the food aspect, while "expensive" and "pricy" may be used to describe the price aspect. When performing sentiment analysis for the price aspect, "delicious" and "yummy" are noise. By clustering aspect terms into subproblems, our solution is able to learn knowledge of similar aspect terms and exclude noise from the irrelevant adjectives.

2.2.2 Data Augmentation. A key challenge raises from clustering the training data into subproblems is that the data of a subproblem is likely to be unbalanced (e.g., more positive reviews than neutral reviews) which may downgrade the quality of the SVM classifiers. To make each subproblem balanced, we augment the data by upsampling. First, we use the largest class as a reference (e.g., positive class). Then, we aim to increase the number of training instances for the other classes (e.g., neutral and negative classes) until the other classes have the same number of training instances as the referenced class. For increasing the number of training instances of a class (e.g., negative class), we randomly select a subproblem (except the current subproblem) and then randomly choose a training instance of the class (e.g., negative class) in selected subproblem. The sampling process is repeated until all the classes have the same number of training instances. In our solution, whether to use sampling is a learnable parameter for each subproblem.

2.2.3 Feature Customization. After dividing the training data into subsets by clustering, the next important step for our solution is to customize features for each subproblem. The intuition is that the features which are useful in other subproblems may not be useful for the current subproblem. Our key idea is that we rank the features based on their relevance to the current subproblem. The relativity score of a feature f in the i-th subproblem is computed by the following formula based on chi-squared

$$chi(f) = \frac{N(N^i_f N^{\bar{i}}_{\bar{f}} - N^i_{\bar{f}} N^{\bar{i}}_f)^2}{(N^i_f + N^{\bar{i}}_f)(N^i_f + N^i_{\bar{f}})(N^i_{\bar{f}} + N^{\bar{i}}_f)(N^i_{\bar{f}} + N^{\bar{i}}_{\bar{f}})},$$

where N denotes the total number of training instances in the whole problem; N^i_f denotes the number of training instances that have nonzero values in the feature f in the i-th subproblem; $N^i_{\bar{f}}$ denotes the number of training instances that have zero values in the feature f in the i-th subproblem; $N^{\bar{i}}_f$ denotes the number of training instances that have nonzero values in the feature f but not in the i-th subproblem; $N^{\bar{i}}_{\bar{f}}$ is the number of training instances that neither have nonzero values in the feature f nor belong to the i-th subproblem.

2.3 Training and Inference

One important property in our solution is that the feature selector, hyper-plane space identifier and SVM trainer are all learnable components. They can be improved based on the loss obtained from the current SVM classifier. Thus, the SVM classifier trained in our solution is well-tuned for features, hyper-parameters and the parameters in the SVM classifier. Here we elaborate the details of training the model including learning to set the hyper-parameters and training the SVMs.

Learning to Select Features: A learning problem may have many features for the whole problem. Those features may work well for one SVM classifier but poorly for another SVM classifier. It is important that different subproblems use different sets of features, i.e., feature customization. Hence, we need to find out the best feature combination among the features for each SVM classifier. In this paper, our solution ranks the features for all the features, and chooses the best features for each subproblem. For example, the SVM sentiment analysis classifier for the first aspect may use surface features and word similarity features only, while that for the second aspect may use all the features.

Learning to Set the Hyper-Plane Space: The hyper-parameters of SVMs have significant influence on the SVM model quality. The hyper-parameters define the hyper-plane space of the SVMs. In our solution, the hyper-parameters of SVMs are learned rather than manually set. We use the sequential model-based optimization [2] to help select the kernel type and the kernel hyper-parameters for each SVM classifier. The key idea is that we use the history of the hyper-parameters to train a machine learning model which guides the search for the best kernel and its corresponding hyper-parameters. Moreover, our solution also learns to decide whether to use sampling to balance the training instances for the SVMs. After each training instance is represented as a feature vector and the hyper-parameters of the SVMs are set, we train an SVM classifier for each subproblem using ThunderSVM [28].

Inference: Given a test instance $\boldsymbol{x}_t$ with the label y_t, our solution first assigns $\boldsymbol{x}_t$ to the corresponding SVM classifier whose center of the subset of training data is the most similar to y_t. Then the relevant features are selected using the techniques presented in Section 2.2.3, and a label (e.g., positive) is predicted by the SVM classifier.

Table 1: Details of data sets from the LibSVM Website

data sets	cardinality		dimensions	#classes
	training set	test set		
a7a	16,100	16,461	123	2
cod-rna	59,535	271,617	8	2
letter	10,500	5,000	16	26
pendigits	7,494	3,498	16	10

2.4 Time Complexity Analysis for Training

The SVM based solutions generally have lower time complexity than the DNN based solutions. To provide a more concrete example of the time complexity analysis, we provide the time complexity analysis for our proposed solution on the ATSA task, in comparison with a representative solution HAPN [13] which is based on DNNs and achieves high predictive accuracy on ATSA. We denote α as the average sentence length, n as the number of training instances, d as the number of dimensions of the training instances, and t as the training rounds (e.g., the number of epochs). For the deep learning based model (i.e., HAPN), the most time consuming operations are matrix multiplications on Bi-GRU and hierarchical attention [13]. Hence, the time complexity for HAPN is $O(t \cdot n \cdot \alpha \cdot d^3)$, where the matrix multiplication takes $O(d^3)$ for each training instance.

In comparison, the time complexity of SVMs is $O(t \cdot n \cdot d)$ for the SVM training using the Sequential Minimal Optimization algorithm [9]. As we can see, the SVM based solution has a much lower time complexity than the DNN based solution. Note that the number of rounds t and the dimension of training instances in SVMs and neural networks may be different. However, this time complexity analysis provides insights of the training cost of SVMs and neural networks.

3 EXPERIMENTAL STUDY

In this section, we present our experimental study for overall evaluation and our case study on the ATSA task for sentiment analysis. Our proposed solution was implemented in Python and the source code to reproduce our experiments is available in https://github.com/Kurt-Liuhf/absa-svm. The clustering algorithm used in our experiments was k-means. The experiments and case study were conducted on a workstation running Linux with a Xeon E5-2640v4 12 core CPU and 64GB main memory.

3.1 Overall Evaluation

To perform an overall evaluation for our proposed solution, we obtained data sets from the LibSVM Website. The information of the data sets is listed in Table 1. We compare our proposed solution with the single SVM based approach, bagging SVMs and AdaBoost SVMs, and evaluate both the predictive accuracy and efficiency. For fair(er) comparison, kernel and regularization hyper-parameter tuner in our proposed solution was disabled. All the SVM based approaches on all the data sets used the Radial Basis Function (RBF) kernel with γ set to 10, and regularization parameter C set to 5. The results are shown in Tables 2 and 3. As we can see from Table 2, our proposed solution produces predictive accuracy results (in terms of accuracy and F_1 on the test data sets) which are always on the top

Table 2: Accuracy and macro-F_1 comparison with different methods

data sets	accuracy				macro-F_1			
	single SVM	bagging SVM	AdaBoost SVM	ours	single SVM	bagging SVM	AdaBoost SVM	ours
a7a	81.19	81.03	71.31	**82.25**	70.41	70.01	70.19	**73.32**
cod-rna	89.06	89.06	65.88	**89.13**	87.24	87.24	65.36	**87.35**
letter	96.39	95.28	77.60	**96.48**	96.33	95.20	76.91	**96.41**
pendigits	99.01	99.01	90.13	**99.01**	99.01	99.01	90.13	**99.01**

Table 3: Training efficiency of SVM based approaches

data sets	elapsed time (sec)			
	single SVM	bagging SVM	AdaBoost SVM	ours
a7a	145.90	494.86	1271.30	**10.51**
cod-rna	85.66	533.85	1041.21	**16.41**
letter	7.82	56.56	731.78	**4.63**
pendigits	**0.52**	3.98	49.94	0.93

Table 4: Details of the *Laptop* and *Restaurant* data sets

data set		#positive	#negative	#neutral	#total
Laptop	Train	987	866	460	2313
	Test	341	128	169	638
Restaurant	Train	2164	807	637	3608
	Test	728	196	196	1120

among all the SVM based approaches. In terms of training efficiency as shown in Table 3, our solution is the fastest in the larger data sets including *a7a*, *cod-rna* and *letter*. Our solution is 14 to over 100 times faster than other approaches in *a7a*. This is because our proposed solution trains a number of small SVMs, instead of a large SVM classifier which is more computationally expensive. In the *pendigits* data set, the single SVM based approach is slightly faster than ours which is the second fastest approach. This is because the size of *pendigits* is small, and further dividing into subproblems brings extra computation cost.

3.2 Case Study on the ATSA task

To further demonstrate the performance of our proposed solution, we conduct a case study on the aspect term sentiment analysis (ATSA) task. We elaborate the setup of the case study next.

Data sets and dictionaries: We conducted our case study using two popular aspect term sentiment analysis data sets from SemEval 2014 Task 4: *Restaurant* and *Laptop*. The detailed information of the two data sets is listed in Table 4, where "#positive", "#negative", "#neutral" and "#total" denote the number of positive reviews, negative reviews, neutral reviews and the total number of reviews, respectively. In our case study, we used eight sentiment lexicons [11]: (1) *tweet sentiment lexicons*, (2) *hashtag sentiment lexicons* and (3) *sentiment140*, (4) *NRC emotion lexicons*, (5) *Bing Liu's lexicons*, (6) *MPQA subjectivity lexicons*, (7) *Yelp restaurant word–aspect association lexicons* and (8) *Amazon laptop word-aspect association lexicons*. These dictionaries were used for extracting sentiment lexicon features in Section 2.2.3.

Hyper-parameter settings and baselines: The number of subproblems was searched from 1 to 35. The regularization constant, C, of the SVMs was selected from 1 to 2^{20}. The SVM kernel functions considered include Radial Basis Function (RBF), polynomial, Sigmoid and linear kernels. The degree for the polynomial kernel was searched from 1 to 5. The γ term of the RBF kernel was selected from 10^{-3} to 10 divided by the size of the training data set. Which features used in the SVM classifier were automatically learned from the data. We identify the latest solutions to the aspect term sentiment analysis problems, including six latest DNN based solutions without pre-trained models and six latest pre-trained neural models based on BERT. The two groups of baselines are named "Neural Model" and "BERT based Model", respectively, as shown in Table 5.

In our solution, we extracted features including surface features, parse features, word similarity features, and sentiment lexicon features similar to the previous SVM based approach [11]. Those features are dependent on the aspect terms within a subproblem (intra-subproblem), which misses the global review of the problem. To optimize the representation, we extracted aspect term independent features (i.e., inter-subproblem features) by considering information from the other subproblems.

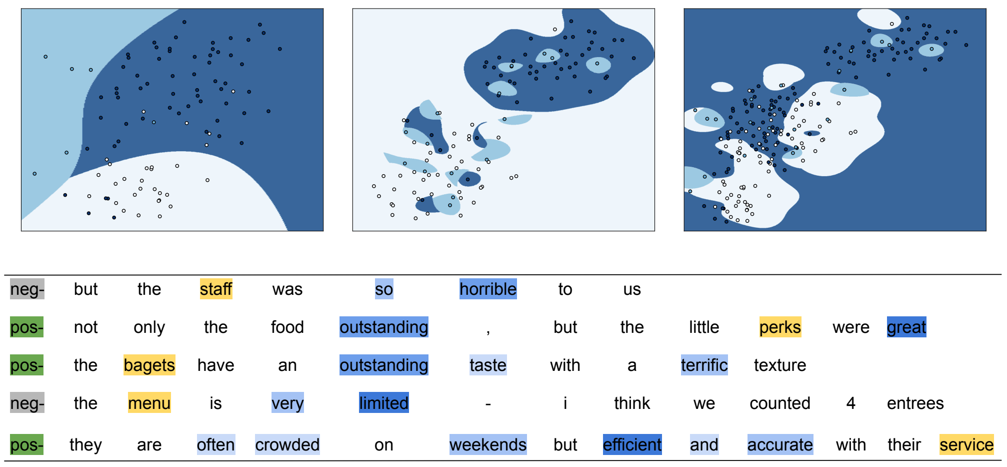
3.2.1 *Accuracy and Macro-F_1 Comparison.*

As we can see from Table 5, our solution outperforms all the DNN based solutions in the "Neural Model" group. When compared with the "BERT based Model" group, our solution outperforms half of them and is competitive to other ones. Our SVM based solution which outperforms all the solutions in the "Neural Model" group is already impressive, because our solution is much simpler while does not exploit pre-trained models or extra labeled data. The comparison between "BERT based Model" and our solution is arguably unfair to our solution, because the SVM model uses little amount of computation resources and training data. We also compare the Macro-F_1 scores among the different methods [23], and the finding is similar. Moreover, our solution consistently outperforms the existing SVM based solution [11] by a large margin (cf. Table 5 bottom). Finally, our solution with multiple SVMs is better than with a single SVM, which confirms the result of Theorem 2.1. For a sanity check, we replaced the SVMs used in our proposed method with BERT. The results are shown in the last row of Table 5, which confirms that enhancing SVMs with problem aware pipeline tends to bring more significant improvement than BERT in our proposed method.

Analysis with Visualization: In order to have a deeper understanding of our solutions, we also performed visualization and looked into the SVM classifiers. As we can see from the top of Figure 2, our SVM base classifiers are able to accurately classify the three classes denoted by dark blue dots, light blue dots and

Table 5: Accuracy and Macro-F$_1$ comparison

Models		Restaurant		Laptop	
		Acc	Macro-F$_1$	Acc	Macro-F$_1$
BERT based Model	LCF-ATEPC [31]	**90.18**	**85.88**	82.29	**79.84**
	LCF-BERT [32]	87.14	81.74	**82.45**	79.59
	BERT-SPC [23]	*84.46*	*76.98*	*78.99*	*75.03*
	SDGCN-BERT [33]	*83.57*	*76.47*	*81.35*	*78.34*
	AEN-BERT [23]	*83.12*	*73.76*	*79.93*	*76.31*
	BERT-PT [30]	*84.95*	*76.96*	*78.07*	*75.08*
Neural Model	HAPN [13]	82.23	-	**77.27**	-
	IMN [7]	**83.89**	**75.66**	75.36	**72.02**
	BILSTM-ATT-G [4]	81.11	72.19	75.44	70.52
	RAM [3]	80.23	70.80	74.49	71.35
	LSTM+SynATT+TarRep [6]	80.63	71.32	71.94	69.23
	PF-CNN [8]	79.20	-	70.06	-
SVM-based Model	existing SVM approach [11]	82.23	73.75	72.27	65.60
	ours (single SVM)	78.57	63.78	72.26	67.61
	ours (multiple SVMs)	**86.79**	**78.81**	**80.25**	**77.07**
Replaced SVMs with BERT	ours (multiple BERTs)	75.98	61.0	62.69	61.0



neg-	but	the	staff	was	so	horrible	to	us						
pos-	not	only	the	food	outstanding	,	but	the	little	perks	were	great		
pos-	the	bagets	have	an	outstanding	taste	with	a	terrific	texture				
neg-	the	menu	is	very	limited	-	i	think	we	counted	4	entrees		
pos-	they	are	often	crowded	on	weekends	but	efficient	and	accurate	with	their	service	

Figure 2: Visualization of the SVM classifier boundaries (top) and important words used for inference (bottom)

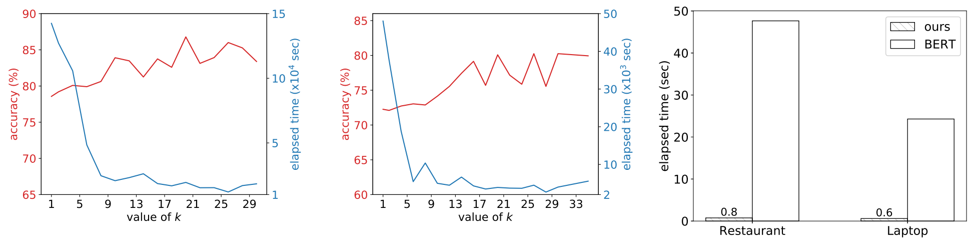
circles. The bottom part of Figure 2 shows the importance (measured by the weight vector) of words used in making a prediction for an aspect term highlighted in yellow. The darker the color, the more important the word is. These results show that the intra- and inter-subproblem features work well and the result is intuitive to interpret. Moreover, we have found that about 80% of the SVM classifiers used data augmentation (cf. Section 2.2).

3.2.2 *Effect of the Number of Subproblems, Inference Efficiency and Model Size Comparison.*

Here, we first study the effect of the number of subproblems on the elapsed time and predictive accuracy. Then, we present inference efficiency and model size comparison.

Effect on varying the number of subproblems: Figure 3a and 3b show the effect of varying the number of subproblems (i.e., k) on accuracy and the elapsed time. As we can see from the figure, the elapsed time for training decreases as the number of subproblems increases. This is because the training cost of each SVM classifier is lower when the subproblem size is smaller. The accuracy tends to increase as the number of subproblems increases. The accuracy hits the highest score when the number of subproblems is 20 for *Restaurant* and 30 for *Laptop*, respectively.

Inference efficiency: Figure 3c shows the batch inference efficiency of BERT and our solution on the two data sets. We used BERT-SPC [23] as an example to study the inference efficiency, as BERT-SPC is open-source and easy-to-use. The batch size used in the experiments was 16 which leads to the best efficiency for BERT on both of the data sets according to our experiments. As we can see from the figure, the efficiency of our solution is over 40 times better than BERT. This is an important property of our solution which



(a) *Restaurant* **(b) *Laptop*** **(c) Inference efficiency**

Figure 3: Effect of # of subproblems and the inference efficiency

produces competitive predictive accuracy to BERT based solutions, while our solution is an order of magnitude faster in inference.

The trained model size: We also calculated the number of parameters in the final trained models. We have found that our solution leads to a much smaller number of parameters than the other models. Specifically, our solution only requires thousands of parameters, while HAPN [13] requires over 2 million parameters and BERT-SPC [23] even requires about 110 million parameters.

4 RELATED WORK

Here we review work on SVMs and aspect term sentiment analysis.

4.1 Ensemble SVMs

A common way to improve machine learning algorithms is to use ensemble models [22, 27]. Many existing studies tried to use ensemble models to improve SVMs. There are two main approaches of ensembling SVMs: bagging and boosting [17]. To improve the classification performance of the SVMs, an ensemble SVM model with bagging was proposed by Kim et al. [10]. Each base SVM is trained independently with the randomly chosen training instances by a bootstrap technique. Then, the trained classifiers are used altogether to make a collective decision based on techniques such as the majority voting. Mordelet and Vert [19] proposed a bagging-based SVM ensemble model to improve the model performance. The algorithm iteratively trains multiple binary classifiers to discriminate the known positive instances from random subsamples of the unlabeled set, and prediction is the average of the classifiers. Li, Wang and Sung [15] presented a model named AdaBoostSVM which uses SVMs as weak learners in AdaBoost. AdaBoostSVM can adaptively adjust the kernel parameters in SVMs, instead of using a fixed one. Another ensemble SVM with AdaBoost was proposed by the same authors [16], which demonstrates better generality than SVMs on imbalanced classification problems. We compared our solution with bagging and AdaBoost SVMs in our experiments.

4.2 Aspect term sentiment analysis (ATSA)

Aspect term sentiment analysis (ATSA) is a fine-grained sentiment classification task [20]. The key to solving this task highly depends on the extraction of semantic relatedness between aspect terms and their corresponding context. Traditional machine learning methods, including rule based methods and SVM based methods [11], are based on a set of linguistically inspired lexical, semantic and sentiment lexicon features. A series of studies based on DNNs have been dedicated to solving the ATSA task. TD-LSTM [24] adopts a forward and a backward LSTM to model the left and the right contexts of the aspect. TNet [14] stacks a proposed CPT module to fuse the embedding of aspect term into the embedding of whole context words. ATAE-LSTM [26] concatenates aspect embeddings with context word representations and adopts an attention mechanism which lets aspect participate in. TNet-ATT [25] proposes an incremental approach to automatically extract attention supervision information for neural aspect term sentiment classification models. SDGCN-BERT [33] employs Graph Convolutional Networks over the attention mechanism to capture the sentiment dependencies and achieves good predictive accuracy. More recently, large pre-trained language models (e.g., BERT) are used to tackle the ATSA task. The related work includes BERT-PT [30] and BERT-ADA [21]. BERT also be used to extract the word embedding. The examples include AEN-BERT [23] and SDGCN-BERT [33]. Despite their good accuracy, those models contain a large number of parameters (i.e., around 110 million parameters) and are slow in inference.

5 CONCLUSION

In this paper, we have proposed techniques to enhance SVMs with a pipeline which exploits the context of the learning problem. Experimental results have shown that our proposed solution is more efficient, while producing better results than the other SVM based approaches. With the ATSA task as a case study, we have demonstrated that different models including those in SVMs and deep learning have their pros and cons in performance, model size as well as training/inference time. We have demonstrated that an SVM based approach with careful design can achieve competitive predictive accuracy to DNN based approaches. Moreover, our SVM based solution is about 40 times faster in inference and has 100 times fewer parameters than BERT based models. Our research findings can encourage more rethinking on algorithm diversity analysis and evaluation: the community needs to carefully evaluate and design diversified algorithms and models to study their trade-off.

ACKNOWLEDGEMENTS

This research is partially supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-RP-2020-018). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore. Prof. Li is supported by National Natural Science Foundation of China (No.61976062). Prof. Chen is supported by the National Natural Science Foundation of China (Grant No. 62072186), the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2019B1515130001), the Guangzhou Science and Technology Planning Project (Grant No. 201904010197).

REFERENCES

[1] Peter L Bartlett and Shahar Mendelson. 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3, Nov (2002), 463–482.

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*. 2546–2554.

[3] Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 452–461.

[4] Xingyi Cheng, Weidi Xu, Taifeng Wang, and Wei Chu. 2018. Variational Semi-supervised Aspect-term Sentiment Analysis via Transformer. *arXiv preprint arXiv:1810.10437* (2018).

[5] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. 2013. Multi-class classification with maximum margin multiple kernel. In *International Conference on Machine Learning*. 46–54.

[6] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*. 1121–1131.

[7] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2019. An Interactive Multi-Task Learning Network for End-to-End Aspect-Based Sentiment Analysis. *arXiv preprint arXiv:1906.06906* (2019).

[8] Binxuan Huang and Kathleen M Carley. 2018. Parameterized Convolutional Neural Networks for Aspect Level Sentiment Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1091–1096.

[9] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Computation* 13, 3 (2001), 637–649.

[10] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung-Yang Bang. 2002. Support vector machine ensemble with bagging. In *International Workshop on Support Vector Machines*. Springer, 397–408.

[11] Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. 437–442.

[12] Vladimir Koltchinskii, Dmitry Panchenko, et al. 2002. Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics* 30, 1 (2002), 1–50.

[13] Lishuang Li, Yang Liu, and AnQiao Zhou. 2018. Hierarchical Attention Based Position-Aware Network for Aspect-Level Sentiment Analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. 181–189.

[14] Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018. Transformation networks for target-oriented sentiment classification. *arXiv preprint arXiv:1805.01086* (2018).

[15] Xuchun Li, Lei Wang, and Eric Sung. 2005. A study of AdaBoost with SVM based weak learners. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 1. IEEE, 196–201.

[16] Xuchun Li, Lei Wang, and Eric Sung. 2008. AdaBoost with SVM-based component classifiers. *Engineering Applications of Artificial Intelligence* 21, 5 (2008), 785–795.

[17] Richard Maclin and David Opitz. 1997. An empirical evaluation of bagging and boosting. *AAAI/IAAI* 1997 (1997), 546–551.

[18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. *Foundations of machine learning*. MIT press.

[19] Fantine Mordelet and J-P Vert. 2014. A bagging SVM to learn from positive and unlabeled examples. *Pattern Recognition Letters* 37 (2014), 201–209.

[20] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. 27––35.

[21] Alexander Rietzler, Sebastian Stabinger, Paul Opitz, and Stefan Engl. 2019. Adapt or Get Left Behind: Domain Adaptation through BERT Language Model Finetuning for Aspect-Target Sentiment Classification. *arXiv preprint arXiv:1908.11860* (2019).

[22] Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 4 (2018), e1249.

[23] Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. 2019. Attentional Encoder Network for Targeted Sentiment Classification. *arXiv preprint arXiv:1902.09314* (2019).

[24] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective LSTMs for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100* (2015).

[25] Jialong Tang, Ziyao Lu, Jinsong Su, Yubin Ge, Linfeng Song, Le Sun, and Jiebo Luo. 2019. Progressive Self-Supervised Attention Learning for Aspect-Level Sentiment Analysis. *arXiv preprint arXiv:1906.01213* (2019).

[26] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 606–615.

[27] Zeyi Wen, Hanfeng Liu, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. 2020. ThunderGBM: Fast GBDTs and random forests on GPUs. *Journal of Machine Learning Research* 21, 108 (2020), 1–5.

[28] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. 2018. ThunderSVM: A fast SVM library on GPUs and CPUs. *The Journal of Machine Learning Research* 19, 1 (2018), 797–801.

[29] Zeyi Wen, Rui Zhang, Kotagiri Ramamohanarao, and Li Yang. 2018. Scalable and fast SVM regression using modern hardware. *World Wide Web* 21, 2 (2018), 261–287.

[30] Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis. *arXiv preprint arXiv:1904.02232* (2019).

[31] Heng Yang, Biqing Zeng, JianHao Yang, Youwei Song, and Ruyang Xu. 2019. A Multi-task Learning Model for Chinese-oriented Aspect Polarity Classification and Aspect Term Extraction. *arXiv preprint arXiv:1912.07976* (2019).

[32] Biqing Zeng, Heng Yang, Ruyang Xu, Wu Zhou, and Xuli Han. 2019. LCF: A Local Context Focus Mechanism for Aspect-Based Sentiment Classification. *Applied Sciences* 9, 16 (2019), 3389.

[33] Pinlong Zhao, Linlin Hou, and Ou Wu. 2019. Modeling Sentiment Dependencies with Graph Convolutional Networks for Aspect-level Sentiment Classification. *arXiv preprint arXiv:1906.04501* (2019).

A APPENDIX

A.1 Definitions and Theorems

In order to prove Theorem 2.1 in the main text, we first introduce a few definitions below based on common conventions and two theorems [18].

Definition A.1 (Margin $s_h(x, y)$). *A hypothesis is defined based on a function $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, the label associated to point x is the one with the largest score $h(x, y)$, the margin $s_h(x, y)$ of the function h at a labeled instance (x, y) is*

$$s_h(x, y) = h(x, y) - \max_{y' \neq y} h(x, y')$$

Definition A.2 (Margin loss function). *For any $\rho > 0$, the ρ-margin loss is the function $L_\rho : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ defined for all $y, y' \in \mathbb{R}$ by $L_\rho(y, y') = \ell_\rho(yy')$ with*

$$\ell_\rho(v) = min\left\{1, max(0, 1 - \frac{v}{\rho})\right\} = \begin{cases} 1, & if\ v \leq 0 \\ 1 - \frac{v}{\rho}, & if\ 0 \leq v \leq \rho \\ 0, & if\ \rho \leq v \end{cases}$$

It is similar to hing loss where $\ell_\rho(v)$ decreases linearly from 1 to 0.

Definition A.3 (Empirical margin loss). *Given a sample $S = \{z_1 = (x_1, y_1), \ldots, z_m = (x_m, y_m)\}$ and a hypothesis h, the empirical margin loss is defined by $\hat{R}_{S,\rho}(h) = \frac{1}{m} \sum_{i=1}^{m} \ell_\rho(s_h(x_i, y_i))$.*

Definition A.4 (Empirical Rademacher complexity [1]). *We use $\mathcal{H}$ to denote a hypothesis set, and define a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a set $\mathcal{G} = \{g : (x, y) \to L(h(x), y) : h \in \mathcal{H}\}$. Furthermore, we represent $\mathcal{G}$ as a family of functions mapping from $\mathcal{Z}$ to $[a, b]$ and $S = \{z_1, \ldots, z_m\}$ is a fixed sample of size m with instances in $\mathcal{Z}$. The empirical Rademacher complexity of $\mathcal{G}$ with respect to the sample S is given by $\hat{\mathfrak{R}}_S(\mathcal{G}) = \mathbb{E}_{\sigma}\left[sup_{g \in \mathcal{G}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(z_i)\right]$,* *where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^\top$ and σ_i is a independent uniform random variable taking value in $\{-1, +1\}$ with equal probability.*

Definition A.5 (Rademacher complexity). *Let $\mathcal{D}$ be the distribution from which instances are drawn. For any number $m \geq 1$, the Rademacher complexity of $\mathcal{G}$ is the expectation of the empirical Rademacher complexity over all the samples of size m drawn from $\mathcal{D}$: $\mathfrak{R}_m(\mathcal{G}) = \underset{S\sim\mathcal{D}^m}{\mathbb{E}}[\hat{\mathfrak{R}}_S(\mathcal{G})]$, where $S \sim \mathcal{D}^m$ means S consists of m instances drawn from $\mathcal{D}$.*

With the above definitions, we can introduce the generalization bound for multi-class classification [12].

Theorem A.1 (Margin bound for multi-class classification). *Let $\mathcal{H} \subseteq \mathbb{R}^{\mathcal{X}\times\mathcal{Y}}$ be a hypothesis set with $\mathcal{Y} = \{1, 2, \ldots, k\}$. We define $\Pi(\mathcal{H}) = \{x \longrightarrow h(x, y) : y \in \mathcal{Y}, h \in \mathcal{H}\}$ and $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following multi-class classification generalization bounds holds for all $h \in \mathcal{H}$:*

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{4k}{\rho}\mathfrak{R}_m(\Pi(\mathcal{H})) + \sqrt{\frac{log\frac{1}{\delta}}{2m}},$$

The generalization bound for multi-class classification can be computed, when considering the kernel based hypotheses in SVMs [5].

Theorem A.2 (Margin bound for multi-class classification with kernel-based hypotheses). *Let $K : \mathcal{X}\times\mathcal{X} \longrightarrow \mathbb{R}$ be a positive definite symmetric (PDS) kernel and let $\Phi : \mathcal{X} \longrightarrow \mathbb{H}$ be a mapping related to K. Assume that there exists $r > 0$ such that $K(x, x) \leq r^2$ for all $x \in \mathcal{X}$. We define the kernel-based hypothesis space as $\mathcal{H}_{K,p} = \{(x, y) \in \mathcal{X} \times \mathcal{Y} \longrightarrow \boldsymbol{w}_y \cdot \Phi(x) : \boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k)^\top, ||\boldsymbol{W}||_{\mathbb{H},p} = (\sum_{l=1}^{k} ||\boldsymbol{w}_l||_{\mathbb{H}}^{p})^{1/p} = \left(\sum_{l=1}^{k} (\sqrt{\boldsymbol{w}_l \times \boldsymbol{w}_l})^p\right)^{1/p} \leq \Lambda$ for any $p \geq 1$ and $\Lambda > 0$ is the upper bound of the norm of the above hypothesis set$\}$. Given $\rho > 0$, then for any $\delta > 0$, with probability at least $1 - \delta$, $R(h) \leq \hat{R}_{S,\rho}(h) + 4k\sqrt{\frac{r^2\Lambda^2/\rho^2}{m}} + \sqrt{\frac{log\frac{1}{\delta}}{2m}}$ holds for all $h \in \mathcal{H}_{K,p}$.*

According to Theorem A.2, the generalization bound for multi-class kernel SVMs can be rewritten as follow.

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m}\xi_i + 4k\sqrt{\frac{r^2\Lambda^2}{m}} + \sqrt{\frac{log\frac{1}{\delta}}{2m}},$$

where $\rho = 1$ and $\hat{R}_{S,\rho}(h)$ in Theorem A.2 can be expressed using hinge loss with $\xi_i = max\{1 - [\boldsymbol{w}_{y_i} \cdot \Phi(x_i) - max_{y' \neq y_i} \boldsymbol{w}_{y'} \cdot \Phi(x_i)], 0\}$.

B PROOF OF THEOREM 2.1

Given the definitions and theorems above, we can now prove Theorem 2.1 as follow.

Proof. Let $S = (z_1, \ldots, z_m)$ denote a sample of size m. For all $l \in \{1, 2, ..., k\}$, the inequality $||\bar{\boldsymbol{w}}_l||_{\mathbb{H}} \leq (\sum_{l=1}^{k} ||\bar{\boldsymbol{w}}_l||_{\mathbb{H}}^{p})^{1/p} = ||\bar{\boldsymbol{W}}||_{\mathbb{H},p}$ always holds. Since $||\bar{\boldsymbol{W}}||_{\mathbb{H},p} \leq (\sum_{l=1}^{k} ||\bar{\boldsymbol{w}}_{*,l}||_{\mathbb{H}}^{p})^{1/p} \leq \bar{\Lambda}$ where $\bar{\Lambda} > 0$, $\bar{\boldsymbol{w}}_{*,l} = max\{\bar{\boldsymbol{w}}_{1,l}, \bar{\boldsymbol{w}}_{2,l}, ..., \bar{\boldsymbol{w}}_{c,l}\}$ and c is the number of chunks, we have $||\bar{\boldsymbol{w}}_l||_{\mathbb{H}} \leq \bar{\Lambda}$ for all $l \in \{1, 2, ..., k\}$. The representation of $\bar{\boldsymbol{w}}_l$ is as follows.

$$\bar{\boldsymbol{w}}_l = \begin{cases} \bar{\boldsymbol{w}}_{1,l}, & \text{if all } x \text{ meets condition } 1 \\ \bar{\boldsymbol{w}}_{2,l}, & \text{if all } x \text{ meets condition } 2 \\ \quad\ldots & \\ \bar{\boldsymbol{w}}_{c,l}, & \text{if all } x \text{ meets condition } c \end{cases}$$

where $\bar{\boldsymbol{w}}_{i,l}$ corresponds to the weight vector of the i-th multi-class SVM, and is determined by the corresponding sample set $S_i = \{z : x \text{ meets condition } i\}$ for all $i \in \{1, 2, ..., c\}$. Concretely, S_i can denote a sample where all the instances in the i-th subproblem of the ATSA task. If the constraint "if all x meets condition i" is satisfied, then $\bar{\boldsymbol{w}}_l$ equals to $\bar{\boldsymbol{w}}_{i,l}$ (e.g., S_i belongs to the i-th subproblem and the i-th multi-class SVM is responsible for the subproblem).

According to Theorem A.1, the key step of the proof lies in bounding the term $\mathfrak{R}_m(\Pi(\bar{\mathcal{H}}_{K,p}))$. We have

$$\mathfrak{R}_m(\Pi(\bar{\mathcal{H}}_{K,p})) = \frac{1}{m}\underset{S\sim\mathcal{D}^m,\boldsymbol{\sigma}}{\mathbb{E}}\left[\underset{\substack{y\in\mathcal{Y}\\ ||\bar{\boldsymbol{W}}||<\bar{\Lambda}}}{sup}\left\langle \bar{\boldsymbol{w}}_y, \sum_{i=1}^{m}\sigma_i\Phi(x_i)\right\rangle\right]$$

$$\leq \frac{1}{m}\underset{S\sim\mathcal{D}^m,\boldsymbol{\sigma}}{\mathbb{E}}\left[\underset{\substack{y\in\mathcal{Y}\\ ||\bar{\boldsymbol{W}}||<\bar{\Lambda}}}{sup}||\bar{\boldsymbol{w}}_y||_{\mathbb{H}}\left\|\sum_{i=1}^{m}\sigma_i\Phi(x_i)\right\|_{\mathbb{H}}\right]$$

$$\leq \frac{\bar{\Lambda}}{m}\underset{S\sim\mathcal{D}^m,\boldsymbol{\sigma}}{\mathbb{E}}\left[\left\|\sum_{i=1}^{m}\sigma_i\Phi(x_i)\right\|_{\mathbb{H}}\right] \leq \frac{\bar{\Lambda}}{m}\left[\underset{S\sim\mathcal{D}^m,\boldsymbol{\sigma}}{\mathbb{E}}\left[\left\|\sum_{i=1}^{m}\sigma_i\Phi(x_i)\right\|_{\mathbb{H}}^2\right]\right]^{1/2}$$

$$\leq \frac{\bar{\Lambda}}{m}\left[\underset{S\sim\mathcal{D}^m,\boldsymbol{\sigma}}{\mathbb{E}}\left[\sum_{i=1}^{m}||\Phi(x_i)||_{\mathbb{H}}^2\right]\right]^{1/2} \leq \frac{\bar{\Lambda}\sqrt{mr^2}}{m}$$

□

B.1 Tighter Bound than Multi-class Classification with Single SVM

We can represent the weight vector $\boldsymbol{w}_l$ in the multi-class classification with single SVM in the piecewise form as follows.

$$\boldsymbol{w}_l = \begin{cases} \boldsymbol{w}_{1,l}, & \text{if all } x \text{ meets condition } 1 \\ \boldsymbol{w}_{2,l}, & \text{if all } x \text{ meets condition } 2 \\ \quad\ldots & \\ \boldsymbol{w}_{c,l}, & \text{if all } x \text{ meets condition } c \end{cases}$$

where $\boldsymbol{w}_{1,l} = \boldsymbol{w}_{2,l} = ... = \boldsymbol{w}_{c,l}$. We have the inequality $||\bar{\boldsymbol{w}}_{i,l}||_{\mathbb{H}} \leq ||\boldsymbol{w}_{i,l}||_{\mathbb{H}}$, since $\bar{\boldsymbol{w}}_{i,l}$ represents the optimal hyperplane with a larger margin than $\boldsymbol{w}_{i,l}$. Thus, we have $||\bar{\boldsymbol{w}}_l||_{\mathbb{H}} \leq ||\boldsymbol{w}_l||_{\mathbb{H}}$, and

$$\bar{\xi}_i = max\{1 - [\bar{\boldsymbol{w}}_{y_i} \cdot \Phi(x_i) - max_{y' \neq y_i}\bar{\boldsymbol{w}}_{y'} \cdot \Phi(x_i)], 0\} \leq \xi_i, \quad (2)$$

for all $i \in \{1, 2, ..., m\}$. Generally, the hyperplane determined by the samples would not satisfy the equality that $||\bar{\boldsymbol{w}}_l||_{\mathbb{H}} = ||\boldsymbol{w}_l||_{\mathbb{H}}$. Since $\bar{\boldsymbol{w}}_{*,l}$ satisfies $||\bar{\boldsymbol{w}}_{*,l}||_{\mathbb{H}} = max\{||\bar{\boldsymbol{w}}_{1,l}||_{\mathbb{H}}, \ldots, ||\bar{\boldsymbol{w}}_{c,l}||_{\mathbb{H}}\}$, and $||\bar{\boldsymbol{W}}||_{\mathbb{H},p} = (\sum_{l=1}^{k} ||\bar{\boldsymbol{w}}_l||_{\mathbb{H}}^{p})^{1/p} \leq (\sum_{l=1}^{k} ||\bar{\boldsymbol{w}}_{*,l}||_{\mathbb{H}}^{p})^{1/p} \leq \bar{\Lambda}$, the following inequality always holds.

$$||\bar{\boldsymbol{W}}||_{\mathbb{H},p} \leq \bar{\Lambda} \leq (\sum_{l=1}^{k} ||\boldsymbol{w}_l||_{\mathbb{H}}^{p})^{1/p} = ||\boldsymbol{W}||_{\mathbb{H},p} \leq \Lambda. \quad (3)$$

As $\bar{\Lambda} \leq \Lambda$ and $\bar{\xi}_i \leq \xi$ and $\bar{\Lambda}$ is a lower bound of $(\sum_{l=1}^{k} ||\boldsymbol{w}_l||_{\mathbb{H}}^{p})^{1/p}$, therefore our bound shown in Theorem 2.1 is tighter than the bound shown in Theorem 1.2.