# Online Multiscale-Data Classification Based on Multikernel Adaptive Filtering with Application to Sentiment Analysis

Ran Iwamoto[1], Masahiro Yukawa[1,2]
1. Department of Electronics and Electrical Engineering, Keio University, Japan
2. Center for Advanced Intelligence Project, RIKEN, Japan

***Abstract*—We present an online method for multiscale data classification, using the multikernel adaptive filtering framework. The target application is Twitter sentiment analysis, which is a notoriously challenging task of natural language processing. This is because (i) each tweet is typically short, and (ii) domain-specific expressions tend to be used. The efficacy of the proposed multiscale online method is studied with dataset of Twitter. Simulation results show that the proposed approach achieves a higher F1 score than the other online-classification methods, and also outperforms the nonlinear support vector machine.**

***Index Terms*—reproducing kernel, sentiment analysis, online learning**

## I. Introduction

This paper addresses the task of classifying a vast amount of customer reviews of products or services as positive or negative opinions. This machine learning task is often called *sentiment analysis*, and it has gained significant attention over the years. In particular, the specific social networking service of Twitter has given measurable impacts on society including politics, but it is challenging to analyze the polarity of emotion from tweets posted to Twitter. A major reason is that tweets tend to include specific expressions (words) depending on generations, communities, which are changing over time. Also, each tweet typically contains a few sentences (or often a few words), which could be grammatically incorrect.

Classification performance is a function of three factors: *preprocessing*, *feature extraction*, and *classifier*. The existing preprocessing methods include the morphological analysis method (consisting of two methods: word segmentation and part-of-speech tagging) [1], the Porter stemming algorithm (replacing inflectional forms into their roots) [2], stopwords [3], [4] (deleting irrelevant words such as *the, a* and *I*), to name a few. In SemEval-2013, NRC-Canada won the 1st prise; the best score was achieved based on bag-of-words [5] that were created elaborately and manually by natural language processing's experts; the feature vectors were then classified by the simple-linear support vector machine [6]. The latter one considers the semantic similarity of words and documents, yielding better performance than the former one [7], [8]. Although these word representation models such as word2vec [9]–[11] are created semantically, they cannot capture sentiment-polarity information adequately. This is because words with opposite sentiment polarities may be represented similar vectors; this is called the good-bad problem [12], [13]. To embed sentiment information in word representations, several methods have been proposed such as word refining [14]. For over 60 years, a lot of efforts have been devoted to devising sophisticated preprocessing [15] and feature-extraction techniques in the natural language processing community.

The importance of online learning is increasing due to the rapid growth of data volume. Incremental semantic analysis has become possible [16]. Although long short-term memory [17] is a novel online learning method and has achieved high performances, there are few works (if any) which process NLP data sequentially and focus on the order of input data itself.

The developed preprocessing and feature-extraction methods enhance the classification performance, the F1 score reported suggests that the performance needs to be improved further. Our research question is the following: is it possible to enhance the classification performance of the Twitter data (which is notoriously difficult to analyze) by using a classifier based on the state-of-the-art nonlinear technique? Also, it is of great interest to develop an *online* classifier that updates the classification boundary incrementally due to several reasons including the large volume of data and the sequential nature as well as the evolving words in Twitter (see Section 4).

In this paper, we investigate the use of multikernel adaptive filtering [18], [19] in sentiment analysis. Specifically, each datum (tweet) is transformed into a feature vector by a distributed representation model, and those feature vectors are classified based on multikernel adaptive filtering. The classification boundary is updated iteratively with training data which are supposed to arrive sequentially. We use the data employed in SemEval-2013 competition (see Section 3.1), and compare the performance (F1 score) of our approach to the existing online- and batch approaches. Publicly available pretrained feature vectors based on distributed representation are employed for our approach because these dimensions are relatively small (25–300). To measure the performance of our multiscale online classifier, we apply our method to Twitter sentiment classification task (positive and negative). The proposed approach achieves the Macro-F1 score 0.8332,

This work was supported by KAKENHI 18H01446, 15H02757.

978-9-0827-9703-9/19/$31.00 ©2019 IEEE

while the best score of the existing method was 0.7814. Multiscaleness of the proposed approach is discussed based on the experimental results presented in the paper.

**Notation:** The sets of all real numbers, nonnegative integers, and positive integers are denoted by $\mathbb{R}$, $\mathbb{N}$ and $\mathbb{N}^*$, respectively. The transpose of vector/matrix is denoted by $(\cdot)^{\mathrm{T}}$. The inner product and the norm are defined by $\langle \boldsymbol{a}, \boldsymbol{b} \rangle := \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$, $\|\boldsymbol{a}\| := \sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle}$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$, $p \in \mathbb{N}^*$.

## II. Online Classification Approach Based on Multikernel Adaptive Filtering

We consider the situations in which a sequence $(\boldsymbol{x}_n)_{n\in\mathbb{N}} \subset \mathcal{U}$ of input vectors (samples) and a sequence $(y_n)_{n\in\mathbb{N}} \subset \mathbb{R}$ of outputs (labels) arrive in a sequential manner, where $\mathcal{U} \subset \mathbb{R}^L$, $L \in \mathbb{N}^*$, is the input space. The goal of the online classification task is then to find a classifier that discriminates those sequential data as accurately as required. In the following, we consider the binary classification case.

Let us consider the binary case when the labels are $+1$ or $-1$; i.e., $y_n \in \{+1, -1\}$ (positive and negative). In this case, the goal of classification is to find a nonlinear function $\varphi : \mathcal{U} \to \mathbb{R}$ that discriminates test data $(\boldsymbol{x}_n^{\mathrm{test}}, y_n^{\mathrm{test}})$ in such a way that $\varphi(\boldsymbol{x}_n^{\mathrm{test}}) > 0$ if $y_n^{\mathrm{test}} = +1$, and $\varphi(\boldsymbol{x}_n^{\mathrm{test}}) < 0$ if $y_n^{\mathrm{test}} = -1$.

**Multikernel adaptive filtering model.** Our classifier based on multikernel adaptive filter is given by [18]

$$\varphi_n(\boldsymbol{x}) = \sum_{q\in\mathcal{Q}} \sum_{j\in\mathcal{J}_{q,n}} \alpha_{j,n}^{(q)} \kappa_q(\boldsymbol{x}, \boldsymbol{x}_j), n \in \mathbb{N}, \boldsymbol{x} \in \mathcal{U}, \tag{1}$$

where $\alpha_{j,n}^{(q)} \in \mathbb{R}$ is the coefficient to be adjusted in online fashion, $\mathcal{J}_{q,n} \subset \{0, 1, 2, \ldots, n\}$, and

$$\kappa_q(\boldsymbol{x}, \boldsymbol{y}) := \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2{\sigma_q}^2}\right), \ \boldsymbol{x}, \boldsymbol{y} \in \mathcal{U}, \tag{2}$$

where $q \in \mathcal{Q} := \{1, 2, \cdots, Q\}$ are Gaussian kernels with different kernel parameters $\sigma_1 > \sigma_2 > \cdots > \sigma_Q > 0$. The set $\mathcal{D}_{q,n} := \{\kappa_q(\cdot, \boldsymbol{x}_j)\}_{j\in\mathcal{J}_{q,n}}$, $n \in \mathbb{N}$, is called *dictionary* for the $q$th kernel, and it needs to be constructed in online fashion as well.

The update of the multikernel adaptive filter has two steps: (i) the dictionary updating and (ii) the coefficient updating.

**Step 1: dictionary updating.** The basic idea of our dictionary growing strategy is the following.

1) Make a smooth boundary using the coarsest (largest-scale) kernel $\kappa_1$.
2) Express locally intricate structures of the boundary using the second coarsest kernel $\kappa_2$.
3) Express finer local structures using $\kappa_3$, and so on.

Let us now present the specific procedure. When the new datum $\boldsymbol{x}_n$ arrives, the following error condition is checked first:

$$\max\{1 - y_n\varphi_n(\boldsymbol{x}_n), 0\} > \epsilon \quad \text{(large hinge loss)} \tag{3}$$

for some small constant $\epsilon \in (0, 1)$. Here, the left side of (3) is the hinge loss function widely used for classification [20]. If this is unsatisfied, the hinge loss is sufficiently small (i.e., the current datum $\boldsymbol{x}_n$ is classified correctly), and hence there is no need to grow the dictionaries. If the error condition is satisfied, the classification boundary needs to be updated in the vicinity of $\boldsymbol{x}_n$, and hence there is a chance for the new functions $\{\kappa_q(\boldsymbol{x}_n, \cdot)\}_{q\in\mathcal{Q}}$ to enter the dictionaries. To see whether the dictionary corresponding to each scale needs to grow by adding $\kappa_q(\boldsymbol{x}_n, \cdot)$, the following coherence criterion [21] is used for eliminating redundancy from the dictionary:

$$\max_{j\in\mathcal{J}_{q,n}} \left| \frac{\kappa_q(\boldsymbol{x}_n, \boldsymbol{x}_j)}{\sqrt{\kappa_q(\boldsymbol{x}_n, \boldsymbol{x}_n)\kappa_q(\boldsymbol{x}_j, \boldsymbol{x}_j)}} \right| \leq \delta \quad \text{(coherence)} \tag{4}$$

for a prespecified threshold $\delta \in (0, 1)$. This coherence criterion is first applied to the largest-scale kernel. If it is satisfied, $\kappa_1(\boldsymbol{x}_n, \cdot)$ is added into the dictionary so that $\mathcal{D}_{1,n+1} := \mathcal{D}_{1,n} \cup \{\kappa_1(\boldsymbol{x}_n, \cdot)\}$, while all the other dictionaries $\mathcal{D}_{q,n}$, $q = 2, 3, \cdots, Q$, stay the same. If it is unsatisfied, the dictionary $\mathcal{D}_{1,n}$ stays the same (i.e., $\mathcal{D}_{1,n+1} := \mathcal{D}_{1,n}$), and the coherence criterion is applied to the second largest-scale kernel $\kappa_2$. If the criterion is satisfied, $\kappa_2(\boldsymbol{x}_n, \cdot)$ is added into the dictionary so that $\mathcal{D}_{2,n+1} := \mathcal{D}_{2,n} \cup \{\kappa_2(\boldsymbol{x}_n, \cdot)\}$, while all the others remain the same. If it is unsatisfied, the criterion is next applied to $\kappa_3$, and so on.

We finally present our pruning strategy for reducing the dictionary size as well as for avoiding the overfitting problem. The basic idea is to eliminate those dictionary elements that make little contributions to estimation. Specifically, if $|\alpha_{j,n}^{(q)}| \leq \gamma \max\{|\alpha_{i,n}^{(p)}|\}_{i\in\mathcal{J}_{q,n}, p\in\mathcal{Q}}$ for some constant $\gamma \in (0, 1)$, then its corresponding element is removed from the dictionary, and update $\mathcal{D}_{q,n+1}$ accordingly.

**Step 2: coefficient updating.** The coefficients $\alpha_{j,n}^{(q)}$ are updated with the dictionaries updated in Step 1. The coefficient for the possible new entry is set to zero. The output of our estimate is simply rewritten as

$$\varphi_n(\boldsymbol{x}_n) = \boldsymbol{\alpha}_n^{\mathrm{T}}\boldsymbol{k}_n = \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle, \tag{5}$$

where $\boldsymbol{\alpha}_n := [\boldsymbol{\alpha}_{1,n}^{\mathrm{T}} \ \boldsymbol{\alpha}_{2,n}^{\mathrm{T}} \ \cdots \boldsymbol{\alpha}_{Q,n}^{\mathrm{T}}]^{\mathrm{T}}$, $\boldsymbol{k}_n := [\boldsymbol{k}_{1,n}^{\mathrm{T}} \ \boldsymbol{k}_{2,n}^{\mathrm{T}} \cdots \boldsymbol{k}_{Q,n}^{\mathrm{T}}]^{\mathrm{T}}$, $[\boldsymbol{\alpha}_{q,n}]_i := \alpha_{j_i,n}^{(q)}$, $i = 1, 2, \cdots, |\mathcal{D}_{q,n}|$, and $[\boldsymbol{k}_{q,n}]_i := \kappa_q(\boldsymbol{x}_n, \boldsymbol{x}_{j_i})$, $i = 1, 2, \cdots, |\mathcal{D}_{q,n}|$.

We adopt the hinge loss function as a cost function. The coefficient vector is updated as

$$\boldsymbol{\alpha}_{n+1} := \boldsymbol{\alpha}_n + \eta \frac{y_n \max\{1 - y_n \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle, 0\}}{\|\boldsymbol{k}_n\|^2 + \rho} \boldsymbol{k}_n, \tag{6}$$

where $\eta \in [0, 2]$ is the step size and $\rho > 0$ is the regularization parameter. The derivation of (6) is given in the appendix. Intuitively, the coefficient vector is updated with a large step when the output of our estimate has the opposite sign to the training label (as the datum is classified incorrectly in this case). If the sign is the same and the output has its magnitude less than one, the coefficients are updated so that the magnitudes become closer to one.

The multikernel adaptive filter is suitable for classifying multiscale data. Figure 1 illustrates a conceptual diagram of

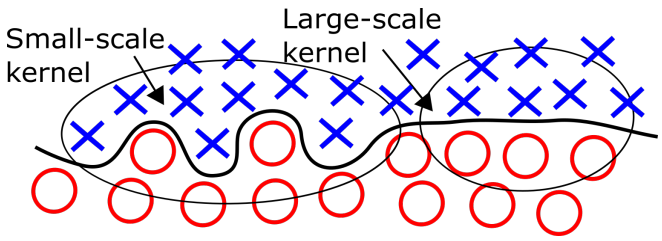


Fig. 1: Multiscale data classification.

TABLE I: The total number of tweets for each polarity available in 2018/2013.

| | Positive | Negative | Neutral |
|---|---|---|---|
| Train | 2643/3662 | 982/1466 | 3446/4600 |
| Dev | 411/575 | 233/340 | 531/739 |
| Test | 1062/1573 | 382/601 | 1176/1640 |

multiscale data classification, where the classification boundary involves locally intricate structures as illustrated in the figure. For such data having a multiscale nature, the large-scale kernel may form a smooth boundary that classifies a large portion of data, while the small-scale kernel may be used to form those finer parts of the boundary to classify those data that cannot be classified correctly by the large-scale kernel. Although there could be a possibility of overfitting if one employs kernels with too small scales, the simultaneous use of different-scale kernels is expected to alleviate it due to the hierarchical way of dictionary growing.

## III. Numerical Examples

We apply the online multiscale-data classification method presented in Section II to the sentiment analysis in Twitter. To compare the performance of the proposed approach and other existing approaches simply, the dataset of tweets are classified into two classes: positive and negative.

### A. Data description and setup

**SemEval-2013 workshop:** We show the effectiveness of multikernel online classification on task 2B "*Message Polarity Classification, Sentiment Analysis in Twitter*[1]" in SemEval-2013 [22], where tweets are classified into three groups: positive, negative, and neutral. SemEval is a competition style workshop held by Association for Computer Linguistics (ACL), and it aims to improve the performance of sentiment analysis tasks and make language resources. Since 2013, polarity analysis tasks in Twitter have been considered in the workshop [22]. The posted time of the oldest tweet in the training data is 13.07.2011 02:18:44.991, and that in the test data is 06.01.2012 22:50:59.046. Indeed, a quarter of the training data were posted earlier than all the test data, while there is no remarkable difference between training and test data in the latest posted time.

**Data download:** The tweet text dataset can be attained at the SemEval-2013 website through Twitter API[2]. Table I shows the number of tweets that were available in 2018 (present study) and 2013 (SemEval-2013 competition). The number of available data varies in time because some tweets may be deleted or become unavailable for several reasons.

**Preprocessing:** We use the text preprocessing methods *ekphrasis* presented by Baziotis [15]. The system was developed for SemEval-2017 Task 4, Sentiment Analysis in Twitter, performing tokenization, word segmentation, word normalization to URLs, hashtags, and user handles.

**Feature extraction:** Classification is conducted based on "tweet vector" which is obtained by summing the word vectors for each tweet. A simple unweighted sum is used since tweets are typically short and weighting makes no much impacts on performance. Glove pretrained word vectors with Twitter dataset [23] of dimension $L = 50$ are used in the following experiments. Two similar vectors in the context-based word representations may have opposite sentiment polarities (the good-bad problem). The pretrained Glove word-vectors are refined using the sentiment refinement model for word embeddings [14].

**Performance measure:** F1-measure [22] is adopted as a performance measure, which is used for the SemEval tasks and is widely accepted in the natural language processing community.

TABLE II: Parameter settings

| method | parameters |
|---|---|
| proposed | $\sigma \in \{20.0, 15.0, 12.0, 10.0, 8.0, 6.0, 4.0, 2.0\}$ |
| | $\gamma \in \{0, 0.001, 0.005, 0.01, 0.05\}$ |
| | $\epsilon \in \{0, 0.001, 0.005, 0.01\}$ |
| | $\eta \in \{0.01, 0.21, 0.41, 0.61, 0.81\}$ |
| | $\rho \in \{0.01, 0.21, 0.41, 0.61, 0.81\}$ |
| | $\delta \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$ |
| SVM (Gaussian) | $\log 2C \in \{-1, 0, \ldots, 3\}$ |
| | $\log 2g \in \{-15, -14, \ldots, 4\}$ |
| online methods | See LIBOL [24]. |

### B. Comparative methods and parameter setting

**Comparative methods:** We compare the performance of the proposed approach to the libSVM [32] as well as some online learning approach including the library LIBOL [24] and the single-kernel counterpart of the proposed approach. The libSVM is simulated as a batch, single-kernel approach. The compared online-classification algorithms in LIBOL are divided into two categories. The first-order algorithms are Perceptron [25], A New Approximate Maximal Margin Classification Algorithm (ALMA) [26], aggressive Relaxed Online Maximum Margin Algorithm (aROMMA) [29], Online Gradient Descent algorithm (OGD) [28]. The second-order algorithms, such as Confidence-Weighted learning algorithm (CW) [27], Narrow Adaptive Regularization Of Weights (NAROW) [30], and Soft Confidence Weighted algorithm-2 (SCW-2) [31], are developed after the development of the first-order algorithms.

**Parameter settings:** The number of kernels is set to $M := 2$. First, we use $k$-means clustering to the train data for determining the kernel-scale parameters approximately. Since the $k$-means for $k = 10$ of the two-point distances of the

[1]https://www.cs.york.ac.uk/semeval-2013/task2.html

[2]API is an abbreviation of Application Programming Interface.

TABLE III: Comparisons to the other approaches for the Glove-based feature vectors.

| method | Macro-F1 | process | parameters |
|---|---|---|---|
| proposed | **0.8332** | online | $\delta = 0.2, \epsilon = 0.001, \eta = 0.41, \rho = 0.21, \sigma_1 = 12.0, \sigma_2 = 8.0, \gamma = 0.001$ |
| single $\sigma = 12.0$ | 0.7461 | online | $\delta = 0.2, \epsilon = 0.001, \eta = 0.41, \rho = 0.21, \gamma = 0.001$ |
| single $\sigma = 8.0$ | 0.7022 | online | $\delta = 0.2, \epsilon = 0.001, \eta = 0.41, \rho = 0.21, \gamma = 0.001$ |
| Perceptron [25] | 0.5172 | online | |
| ALMA [26] | 0.5427 | online | $\eta = 0.8, p = 10, c = 0.0625$ |
| CW [27] | 0.6522 | online | $\eta = 0.6, a = 1$ |
| OGD [28] | 0.7359 | online | $t = 1, c = 1$ |
| aROMMA [29] | 0.7506 | online | |
| NAROW [30] | 0.7641 | online | $c = 0.25, b = 1, a = 1$ |
| SCW-2 [31] | 0.7696 | online | $\eta = 0.9, c = 0.0625, a = 1$ |
| SVM (Gaussian) [32] | 0.7814 | batch | $t = 2$(radial basis function), $\log 2C = 3, \log 2g = -12$ |

training data is approximately 10.0, we select the kernel-scale parameters close to 10.0. The kernel-scale, step-size, coherence, hard-thresholding, and regularization parameters are optimized by grid search within the ranges shown in Table II respectively. For the single-kernel approach, the parameters are selected according to basically the same way as the multi-kernel approach. The parameter choices of the online methods are in accordance with LIBOL [24].

At the end of the learning, the dictionary sizes of the proposed approach for the two kernels were $\mathcal{D}_{1,n} = 1243$ and $\mathcal{D}_{2,n} = 1330$, respectively. For the single kernel method, the dictionary sizes were $\mathcal{D}_n = 1140$, for $\sigma = 12$, and $\mathcal{D}_n = 3133$ for $\sigma = 8$.

The proposed approach is fed with the tweets that are ordered in the posted time.

### C. Results

Table III shows comparisons of the proposed approach to the other methods. Compared to the other online methods, the proposed approach achieves 7-percent higher F1 score. Furthermore, the proposed method attains a higher score than the proposed single-kernel approach and batch single-kernel SVM. The result shows that the multiscaleness of the proposed classifier contributes to finding the reasonable classification boundary. The remarkably high score 0.8332 of the proposed method suggests the potential usefulness of the online sentiment analysis in Twitter.

### D. Discussions

The proposed approach is an online method, while all the other approaches in SemEval-2013 task 2B are batch methods. Online nature is of practical importance because the Twitter data has an online nature due to the following reasons: (i) the tweets are posted randomly, and it is impossible in general to predict when tweets stop arriving and (ii) some new characteristic expressions emerge one after another. The online method processes those streaming data on the fly, and it keeps updating the classification boundary as long as new data keep arriving. Hence, the online method can adapt flexibly to emerging expressions as well as tracking possible changes of trends (e.g., polarity changes of phrases). The onlineness is also highly advantageous in its potential applications to general large-volume data, as it significantly reduces the computational complexity and the memory requirements.

Compared to the single-kernel approach, the multikernel approach attains the significant gain in performance, while the total dictionary size is comparable. To be more precise, the total dictionary size of the multi-kernel approach is smaller than the dictionary size of the single kernel approach for the small-scale kernel. This observation indicates that *the task has multiscale nature* — the sole use of the small-scale kernel makes the dictionary size too large since the kernel scale is too small to estimate the classification boundaries. A large portion of data (tweets) can be discriminated with the large-scale kernel, while some portion of them cannot do due to complicated structures. The small-scale kernel comes into play here to extract those complicated structures efficiently (see Fig. 1).

## IV. Concluding Remarks

We proposed the online method for multiscale data analysis, targeting binary classification tasks in sentiment analysis. The proposed approach was applied to binary classification by using the Twitter dataset used in the SemEval-2013 competition. It achieved the best F1-score 0.8332, while the highest F1-score among the other online approaches was 0.7696 and the F1-score of the batch SVM was 0.7814. The proposed multi-kernel approach also outperformed the single-kernel counterpart of the proposed approach.

The online nature of the proposed approach has significant potential advantages. The process of annotating each datum is made manually by experts and is time-consuming, and thus it may take an unacceptably long time if one waits until all the data are annotated. The proposed online approach enables to perform the annotation and learning processes in parallel, and this permits real-time data analysis. This is clearly desirable for users. In conclusion, the remarkably high score achieved by the proposed approach suggests its potential significance in online classification of multiscale data.

## Acknowledgment

The first author would like to thank Dr. H. Watanabe of IBM Research - Tokyo, Profs. K. Ohara and H. Saito of Keio University for sincere encouragements as well as fruitful discussions on the future scopes.

## APPENDIX: DERIVATION OF (6)

We can assume that $\boldsymbol{k}_n \neq \mathbf{0}$. We show that the update equation (6) for $\rho := 0$ coincides with

$$\boldsymbol{\alpha}_{n+1} := \boldsymbol{\alpha}_n + \eta \left( P_{C_n}(\boldsymbol{\alpha}_n) - \boldsymbol{\alpha}_n \right), \tag{7}$$

where

$$C_n := \{ \boldsymbol{\alpha} \in \mathbb{R}^{r_n} \mid y_n \langle \boldsymbol{\alpha}, \boldsymbol{k}_n \rangle \geq 1 \}. \tag{8}$$

If $\boldsymbol{\alpha}_n \in C_n$ ($\Leftrightarrow 1 - y_n \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle \leq 0$), then $P_{C_n}(\boldsymbol{\alpha}_n) = \boldsymbol{\alpha}_n$. If, on the other hand, $\boldsymbol{\alpha}_n \notin C_n$ ($\Leftrightarrow 1 - y_n \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle > 0$), then the projection onto the halfspace $C_n$ is actually the same as the projection onto the boundary hyperplane

$$H_n := \{ \boldsymbol{\alpha} \in \mathbb{R}^{r_n} \mid y_n \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle = 1 \}. \tag{9}$$

In this case, the projection is given by

$$P_{C_n}(\boldsymbol{\alpha}_n) = P_{H_n}(\boldsymbol{\alpha}_n) = \boldsymbol{\alpha}_n + \frac{y_n(1 - y_n \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle)}{\|\boldsymbol{k}_n\|^2} \boldsymbol{k}_n. \tag{10}$$

Hence, in the general case, the projection can be expressed as

$$P_{C_n}(\boldsymbol{\alpha}_n) = \boldsymbol{\alpha}_n + \frac{y_n \max\{1 - y_n \langle \boldsymbol{\alpha}_n, \boldsymbol{k}_n \rangle, 0\}}{\|\boldsymbol{k}_n\|^2} \boldsymbol{k}_n. \tag{11}$$

Substituting (11) into (7) yields the update equation (6) for $\rho := 0$.

## REFERENCES


[1] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22–31, 1968.

[2] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[3] H. P. Luhn, "A Statistical Approach to Mechanized Encoding and Searching of Literary Information," *IBM Journal of Research and Development*, vol. 1, no. 4, pp. 309–317, Oct. 1957.

[4] R. T.-W. Lo, B. He, and I. Ounis, "Automatically Building a Stopword List for an Information Retrieval System," *Journal of Digital Information Management*, vol. 3, no. 1, pp. 3–8, 2005.

[5] Z. S. Harris, "Distributional Structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954.

[6] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 2. PMLR, Jun. 2014, pp. 1188–1196.

[8] Y. Kim, "Convolutional Neural Networks for Sentence Classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014, pp. 1746–1751.

[9] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 746–751.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *CoRR*, vol. abs/1301.3781, 2013.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[12] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment Embeddings with Applications to Sentiment Analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 2, pp. 496–509, Feb 2016.

[13] S. M. Mohammad, B. J. Dorr, G. Hirst, and P. D. Turney, "Computing Lexical Contrast," *Computational Linguistics*, vol. 39, no. 3, pp. 555–590, 2013.

[14] L.-C. Yu, J. Wang, K. R. Lai, and X. Zhang, "Refining Word Embeddings for Sentiment Analysis," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017, pp. 534–539.

[15] C. Baziotis, N. Pelekis, and C. Doulkeridis, "DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, 2017, pp. 747–754.

[16] N. Kaji and H. Kobayashi, "Incremental Skip-gram Model with Negative Sampling," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, sep 2017, pp. 363–371.

[17] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[18] M. Yukawa, "Multikernel Adaptive Filtering," *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4672–4682, Sep. 2012.

[19] M. Yukawa, "Adaptive Learning in Cartesian Product of Reproducing Kernel Hilbert Spaces," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6037–6048, Nov. 2015.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[21] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online Prediction of Time Series Data With Kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[22] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson, "SemEval-2013 Task 2: Sentiment Analysis in Twitter," in *Proceedings of the Seventh International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2013, pp. 312–320.

[23] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014, pp. 1532–1543.

[24] S. C. Hoi, J. Wang, and P. Zhao, "LIBOL: A Library for Online Learning Algorithms," *Journal of Machine Learning Research*, vol. 15, pp. 495–499, 2014.

[25] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review*, pp. 65–386, 1958.

[26] C. Gentile, "A New Approximate Maximal Margin Classification Algorithm," *Journal of Machine Learning Research*, vol. 2, pp. 213–242, 2001.

[27] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted Linear Classification," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML'08. ACM, 2008, pp. 264–271.

[28] M. Zinkevich, "Online Convex Programming and Generalized Infinitesimal Gradient Ascent," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03. AAAI Press, 2003, pp. 928–935.

[29] Y. Li and P. M. Long, "The Relaxed Online Maximum Margin Algorithm," *Machine Learning*, vol. 46, no. 1-3, pp. 361–387, 2002.

[30] F. Orabona and K. Crammer, "New Adaptive Algorithms for Online Classification," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1840–1848.

[31] J. Wang, P. Zhao, and S. C. H. Hoi, "Exact Soft Confidence-weighted Learning," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. Omnipress, 2012, pp. 107–114.

[32] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.