A Modified Cosine-Similarity based log Kernel for Support Vector Machines in the Domain of Text Classification

Rajendra Kumar Roul
Dept of Computer Science
BITS,Pilani-Goa Campus
Zuarinagar
Goa-403726, India
rkroul@goa.bits-pilani.ac.in

Kushagr Arora
Dept of Computer Science
BITS,Pilani- Goa Campus
Goa-403726, India
kushagrarora786@gmail.com

Ishaan Bansal
Dept of EEE
BITS,Pilani- Goa Campus
Zuarinagar
Goa-403726, India
ishaan.bansal.29@gmail.com

Abstract

The popularity of the internet is increasing day-by-day, which makes tough for the end-user to get desired pages from the web in a short time. Text classification, a branch of machine learning can shed light on this problem. State-of-the-art classifier like Support Vector Machines (SVM) has become very popular in the domain of text classification. This paper studies the effect of SVM using different kernel methods and proposes a modified cosine distance based log kernel *(log cosine)* for text classification which is proved as Conditional Positive Definite (CPD). Its classification performance is compared with other CPDs and Positive Definite (PD) kernels. A novel feature selection technique is proposed which improves the effectiveness of the classification and gathers the crux of the terms in the corpus without deteriorating the outcome in the construction process. From the experimental results, it is observed that CPD kernels provide better results for text classification when used with SVMs compared to PD kernels, and the performance of the proposed log-cosine is better than the existing kernel methods.

Keywords: Classification, Conditional positive definite kernels, Cosine distances, Positive definite kernels, Support Vector Machines

1 Introduction

Text classification plays a vital role in the domain of machine learning where the text data is categorized into different groups of similar data items. Many times the present search engine retrieves invalid links and irrelevant web pages for a submitted user query. This weakens the trust of the user on the search engine and thereby degrade its performance. Text classification, a powerful machine learning technique which categorizes an unseen document into its respective predefined class can help in this direction. Two basic classifications of web pages are there: subject-based and genre-based (Qi and Davison, 2009). In subject-based classification, web pages are classified based on their subject or content. Topic hierarchies of web pages are built by this approach. Web pages in genre-based classification are classified into genre or functional related factors, for example, some web pages genres are "multimedia", "home page", "online transaction", and "news headlines". This classification helps users to find their immediate interest from the web without waiting for a long time. There are many classification techniques that exist in real and can be divided into two broad categories: *eager learner* and *lazy learner*. According to eager learner classification technique, the learner built a classification model when the training dataset is given before it receives the test dataset. It can be thought as if the learning model is ready and eager to classify the new test dataset. Examples of this are decision tree, Bayesian network, support vector machine, rule and association based classifier etc. But in lazy learner classification technique, the things are different. Here, instead of building a classification model, it simply stores the training dataset, hence consumes extra space and after seeing the test dataset, it does the classification based on the similarity to the stored training dataset. Examples include k-nearest neighbors (k-NN) and Cased-based reasoning.

Content and *Context* of the web page play major role during the classification process. The sole content of the page including HTML tags, images, text, videos help for classification. Similarly, the hyperlink present in a web page

Proc. of ICON-2017, Kolkata, India. December 2017 ©*2017 NLPAI*, pages 338–347

also decides the page classification. Binary and multi-class are the two basic types of classification exist for classifying the text documents. Binary classification generally categorizes the documents into one of two pre-defined classes whereas multi-class classification handles more than two classes. Classification again can be either single-label or multi-label which is decided depending on the number of labels that is going to be assigned to a document. Exactly one class label is to be assigned to a document in single-label whereas more than one class label is assigned to a document in multi-label classification. For instance, three-class classification means the classification problem consists of three classes say 'Business', 'Sports' and 'Movies'. Many research works has done in the field of web document classification (Aggarwal and Zhai, 2012)(Qiu et al., 2011)(Sebastiani, 2002) (Roul et al., 2017)(Roul and Sahoo, 2017)(Roul et al., 2016)(Roul and Rai, 2016).

Feature selection plays a major role in text classification because selection of important features not only reduces the training time, but also increases the performance of the classifier by reducing the irrelevant features from the corpus. Further, the algorithms used for feature selection are classified into the following three categories:

i. *Filter methods* (Kira and Rendell, 1992) do not use any classifiers for feature selection instead features are selected on the basis of statistical properties. Hence, these methods are fast to compute and capture the usefulness of the feature set which makes them more practical.

ii. *Wrapper methods* (Kohavi and John, 1997) generate different subsets of features based on some algorithms and test each subset using a classifier. To find the score of feature subsets, wrapper methods use a predictive model, whereas filter methods use a proxy measure.

iii. *Embedded methods* (Lal et al., 2006) combines the advantages of both the above two methods and their computational complexity lies between these two methods.

To make the classification process more efficient, a good classifier is required. From the research, it has been observed that the usage of SVM (Cortes and Vapnik, 1995) in text classification has been largely accurate. Many research works on text classification using SVM kernel has been done in the past (Lodhi et al., 2002)(Tong and Koller, 2001)(Zhang et al., 2008)(Maji et al., 2013). Kernel (a similarity function which takes two input feature vectors and find out how similar they are) boost the performance of SVM especially when the number of training documents is more than the number of keywords/features. Kernel can be used on those algorithms which support the inner product that takes the advantage of the nonlinear mapping of features into a high dimensional space with less computational cost (computing the kernel in a higher dimensional space is easy, but computing the feature vector corresponding to the kernel is computationally expensive). Many researchers have worked on SVM kernel (Hamsici and Martinez, 2009)(Hong et al., 2016)(Ponte and Melko, 2017).

Do SVMs work well for text classification?
The theoretical basis for the good performance of SVMs in classifying text documents is suggested by Jaochims (Joachims, 1998) which establishes the following reasons for the same.

i. *High Dimensional Input Space*: By using overfitting protection, SVM does not depend on the number of features and can able to handle a large volume of feature space.

ii. *Few Irrelevant Features*: In text categorization, getting rid of irrelevant features is not of much help, as most features are relevant for classification. So, one cannot easily overcome the problem of high dimensional input space by getting rid of some irrelevant features.

iii. *Sparse Document Vectors*: It has been shown that SVMs are well suited for classification problems with dense concepts and sparse instances.

In this paper, we studied different existing kernels such as RBF, Linear, Polynomial etc. and compare the classification accuracy of SVMs using those kernels techniques. Boughorbel et. al. (Boughorbel et al., 2005) in their work have shown that using SVMs, Conditionally Positive Definite (CPD) kernels provide more accurate

results than Positive Definite (PD) kernels while classifying images. Knowing that SVMs perform well in text classification, here we aim to show that the performance of SVMs using CPD kernels in classifying text document is better than using normal PD kernels. We propose a new kernel based on cosine distances *(log cosine)*, and shown that it is indeed CPD. A novel feature selection technique is proposed in order to reduce the size of the training feature vector which in turn enhances the performance of the classification process. Experimental results on different benchmark datasets show that the usage of log cosine as a kernel in SVM for text classification is better than the existing kernel methods.

The rest of the paper is organized on the following lines: In Section 2, we have discussed the definitions and background details of the proposed approach. Section 3 discusses the proposed approach followed by the experimental analysis discussed in Section 4. Finally, in Section 5, we concluded the work with some future enhancement.

2 Definitions and Basic Preliminaries

This section discusses few important classes of kernels which includes the proposed modified SVM log kernel based on cosine distance and some other background details that are used in the proposed approach.

Definition 1 (Gram Matrix) *Let $K : X \times X \to \mathbb{R}$ be a kernel function. The matrix with entries $K_{ij} := K(x_i, x_j)$ is called Gram matrix or kernel matrix of K with respect to the patterns $x_1, x_2, \cdots, x_n$.*

Definition 2 (Positive Definite Matrix) *A real $n \times n$ matrix with entries K_{ij} is called positive definite matrix if $\sum_{i,j} c_i c_j K_{ij} \geq 0 \; \forall c_i \in \mathbb{R}$.*

Definition 3 (Positive Definite Kernel) *Let X be a non-empty set and K be a symmetric kernel function defined on $X \times X$. If the matrix with elements $K(x_i, x_j)$ is positive definite $\forall n \in \mathbb{N}$ and all $x_1, x_2, \cdots, x_n \in X$ then K is called positive definite kernel.*

Definition 4 (Conditional Positive Definite Kernel) *Let X be a non-empty set. A symmetric kernel K is called conditionally positive definite if $\sum_i^n \sum_j^n c_i c_j K_{ij} \geq 0$ holds $\forall n \in \mathbb{N}$, $x_1, x_2, \cdots, x_n \in X$ and $c_i \in \mathbb{R}$ with $\sum_{i=1}^n c_i = 0$.*

It is needed to show that the negative squared Euclidean distance kernel i.e. $K(x, y) = -\|x - y\|^2$ is conditional positive definite. The proof directly follows from the definition and can be found in (Cowling, 1983). The negative distance kernel is also known as power kernel or triangular kernel. We adapt another conditionally positive definite kernel called *log kernel*, as a part of the study. The log kernel is defined as

$$K_{\log}(x, y) = -\log\left(1 + \|x - y\|^{\beta}\right)$$

On similar lines of the above result, one can easily show that the log kernel is conditionally positive definite.

2.1 Kernel based on Cosine Distances

After suggesting two Conditionally Positive Definite kernels in negative squared distance and log power kernel, we proceed to explore a new kernel method based on cosine distances. The Cosine distance *(CosDis)* is a variant of the Cosine Similarity *(CosSim)* measure and defined as

$$CosDis = 1 - CosSim$$

where $CosSim = \frac{A \cdot B}{|A||B|}$, A and B are two different instances.

We propose a modified kernel function based on cosine distance. There are valid reasons why one may prefer a kernel based on cosine distance rather than the Euclidean distance. The reason for doing so is mainly due to the specific advantages of cosine similarity measures while classifying the text documents (i.e. when the length of two documents are unequal). It is beneficial to abstract out the magnitude of the term vectors so that one can remove the influence of document length. Documents which are clustered using L2-norm instead of direction (i.e. vectors having different directions can be clustered because of their distances from the origin are similar) are highly susceptible to Euclidean distance. When classifying text documents, one uses the angular distance in order to categorize them by their overall sentiments. Relative frequencies of words in the document and across documents are important. Both of these features are exhibited by cosine distance measures. As a result, we propose a modified SVM log kernel based on cosine distance. The

modified kernel based on the cosine distance is defined using equation 1.

$$K(x, y) = -\log(1 + CosDis(x, y)) \quad (1)$$

Before, we prove the above log kernel is conditionally positive definite, we first state the following theorem (Cowling, 1983).

Theorem 1 *If $K : X \times X \rightarrow (-\infty, 0]$ is conditionally positive definite then the following are conditionally positive definite.*

(i) For each α $(0 < \alpha < 1)$, $-(-K)^{\alpha}$.

(ii) $-\log(1 - K)$.

According to Scholkopf (Scholkopf, 2001):

Theorem 2 *If a kernel K is conditionally positive definite then $K+b$ is conditionally positive definite for any constant $b \in \mathbb{R}$.*

Since $K(x, y) = -CosDis(x, y) = -1 + CosSim(x, y)$ and $CosSim(x, y)$ is positive definite, therefore by Theorem 2, K is conditionally positive definite. By applying Theorem 1, we can prove $K(x, y) := -\log(1 + CosDis(x, y))$ is conditionally positive definite. So our proposed cosine based log kernel is conditionally positive definite.

2.2 Term Frequency and Inverse Document Frequency

Term Frequency (TF) measures how often a term t occurs in a document d whereas Inverse Document Frequency (IDF) measures the importance of t in the entire corpus P. TF-IDF (Sparck Jones, 1972) is a technique which finds the importance of terms in a document based on how they appear in the corpus. The TF-IDF is calculated using equation 2.

$$TF\text{-}IDF_{t,d} = TF_{t,d} \times IDF_t \quad (2)$$

where,

$$TF_{t,d} = \frac{number\ of\ occurance\ of\ t\ in\ d}{total\ length\ of\ d}$$

$$IDF_t = log\Big(\frac{number\ of\ documents\ in\ P}{number\ of\ documents\ contain\ the\ term\ t}\Big)$$

2.3 Cosine-similarity

Cosine-similarity is a technique which measures the similarity between two document vectors ($\overrightarrow{d_1}$ and $\overrightarrow{d_2}$) and can be represented as follows:

$$\textit{cos-sim}(\overrightarrow{d_1}, \overrightarrow{d_2}) = \frac{\overrightarrow{d_1}.\overrightarrow{d_2}}{|\overrightarrow{d_1}| * |\overrightarrow{d_2}|}$$

2.4 Fuzzy C-Means

Fuzzy C-Means (FCM) algorithm (Bezdek et al., 1984) tries to distribute a finite collection of n documents into c clusters. It returns a list of c cluster centroids along with a matrix which shows the degree of membership of each document to different clusters. It aims to minimize the following function:

$$T_m = \sum_{i=1}^{n}\sum_{j=1}^{c} v_{ij}^{m}||d_{ij}||^2$$

where, distance $d_{ij} = x_i - c_j$, m generally set to 2 is the fuzzy coefficient, c_j is the centroid(vector) of cluster j, x_i is the i^{th} document, $v_{ij} \in [0, 1]$ is the degree of membership of x_i with respect to c_j and ($\sum_{j=1}^{c} v_{ji} = 1, i = 1, ..., n$). One can iteratively find the values of c_j and v_{ij} updated with each iteration by using equations 3 and 4.

$$c_j = \frac{\sum_{i=1}^{n} v_{ij}^{m} - x_i}{\sum_{i=1}^{n} v_{ij}^{m}} \quad (3)$$

$$v_{ij} = \frac{1}{\sum_{k=1}^{c}(\frac{||d_{ij}||}{||x_i - c_k||})^{\frac{2}{m-1}}} \quad (4)$$

3 Proposed Methodology

Step 1 *Pre-processing of Documents:*
Consider a corpus having set of classes $(C = c_1, c_2, ..., c_n)$ of documents $(D = d_1, d_2, ..., d_p)$. All documents are pre-processed which includes lexical-analysis, stop-word elimination, stemming, and index terms extraction. The term-document matrix is constructed using the vector space model, where TF-IDF value is used to measure the weight of the term t_i in its respective document d_j and is shown in the Table 1.

Table 1: Term-document matrix

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
t_r	t_{r1}	t_{r2}	t_{r3}	...	t_{rp}

Step 2 *Clusters formation*:
The entire corpus is clustered to generate the groups of similar terms. For this purpose, traditional Fuzzy C-means clustering algorithm (MacQueen and others, 1967) is applied on the term-document matrix of the corpus which generates 's' term-document clusters $td = \{td_1, td_2, ..., td_s\}$ having each td_i of dimension $b \times p$, where b is the number of terms and is shown in the Table 2.

Table 2: Reduce term-document matrix

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
t_b	t_{b1}	t_{b2}	t_{b3}	...	t_{bp}

Step 3 *Top features selection from each cluster:*
Top features are selected from each term-document cluster td_i using the following steps:

i. *Computing the cosine-similarity*:
The centroid of td_i is computed using equation 5.

$$\overrightarrow{sc_i} = \frac{\sum_{j=1}^{r} \overrightarrow{t_i}}{r} \quad (5)$$

where sc_i is the centroid of td_i. Next, the cosine-similarity score between each term $t_j \in td_i$ and sc_i is computed using equation 6.

$$cos\text{-}sim(\overrightarrow{t_j}, \overrightarrow{sc_i}) = \frac{\overrightarrow{t_j}.\overrightarrow{sc_i}}{|\overrightarrow{t_i}| * |\overrightarrow{sc_i}|} \quad (6)$$

ii. *Generating synonym list*:
Select a term $t_j \in td_i$ randomly and store its synonyms using WordNet[1] in a file $synonym\_list_{old}$ which will constitute the synonym list for t_j (example shown in Table 3). Find the common terms between $synonym\_list_{old}$ and td_i, and add them to a new synonym list called $synonym\_list_{new}$ and discard them from td_i so that the synonyms of t_j will be no longer in td_i as they are already present in the new synonym list ($synonym\_list_{new}$) of t_j. Remove the old synonym list ($synonym\_list_{old}$) of t_j. Repeat this step for the remaining word of td_i till it get exhausted.

iii. *Constructing feature vector*:
Now for every randomly selected term t_j, there is a corresponding new synonym list ($synonym\_list_{new}$) contain its synonym terms. Next from each $synonym\_list_{new}$ of t_j, select the top m% terms having high cosine-similarity scores and merge them to an array IFV which will constitute the reduced feature vector of td_i. The detail discussion are shown in Algorithm 1.

Table 3: Synonym list of terms

Term	Synonym list
Explain	account for, clarify, define, elaborate, interpret, justify,
Fast	expeditiously, fleet, hastily, hasty, quickly, mercurial, quick, rapid, speedy, snappy, swiftly, rapidly, snappily, speedily, posthaste, like a flash
File	directory, data, case, book, folder, list, information, register, repository, charts, documents, cabinet
Index	pointer, mark, needle, indicator, ratio, rule, symbol, formula, token
Program	plan, schedule, curriculum, bill, syllabus, record, timetable, bulletin, arrangements
Thesaurus	glossary, lexicon, terminology, vocabulary, reference book, source book

Step 4 *Generating the training feature vector:*
Repeat step 3 for all term document cluster td_i and merge the terms of each IFV into a final array RFV after removing all the duplicate terms. Now RFV is the required reduced training feature vector used for classification.

Step 5 *Training SVM on reduced feature vector:*
The SVM classifier is trained using the train-

[1]http://wordnet.princeton.edu/

Algorithm 1: Top feature selection

Data: Cluster td_i having cosine-similarity values of each term t_j
Result: Final feature vector (RFV) of td_i
$Term\_List(TL) \leftarrow \phi$
$Synonym\_List_{t_j}(SL_{t_j}) \leftarrow \phi$
$New\_Synonym\_List_{t_j}(NSL_{t_j}) \leftarrow \phi$
$Extra\_List(EL) \leftarrow \phi$ //A two dimensional list
$TL \leftarrow$ terms of td_i
for *each term* $t_j \in$ *TL (selected randomly)* **do**
  $SL_{t_j} \leftarrow$ all the synonyms of t_j found in Wordnet
  for *each term* $t_k \in TL$ **do**
    // except $\{t_j\}$
    if t_k *present in* SL_{t_j} **then**
      add t_k to the NSL_{t_j} of t_j and drop t_k from TL
    end
  end
  $EL \leftarrow EL \cup NSL_{t_j}$ //merge the synonym required list of t_j to EL
  $NSL_{t_j} \leftarrow \phi$
  $SL_{t_j} \leftarrow \phi$
end
for *each* $NSL_{t_j} \in EL$ **do**
  select the top m% terms T having highest cosine-similarity values from NSL_{t_j}
  $RFV \leftarrow RFV \cup T$
end
return RFV

ing feature vector (RFV) on positive definite kernels such as linear, RBF, polynomial, and sigmoid kernels. Following this, the SVM is trained using conditionally positive definite kernels such as Negative Euclidean, log Power kernels, and proposed log cosine kernel.

4 Experimental Section

Four benchmark datasets are used for experimental work (DMOZ[2], 20-Newsgroups[3], Reuters, Classic3[4] and WebKB[5]). The details of these datasets are discussed below:

4.1 20-Newsgroups

20-Newsgroups is a standard machine learning dataset and it has 11293 training and 7528 testing documents classified into 20 classes. Three classes are taken into consideration (*alt.atheism*, *soc.religion.christian* and *misc.forsale*) for experimental purpose, consisting of total 1663 training and 1107 test documents. Total number of terms used is 20422 and among them, 16270 are used for training.

4.2 DMOZ

DMOZ is one of the largest dictionaries on the Web. It has 14 categories out of which 3 categories (*Arts, Homes and Science*) of 5238 documents are used for experimental purpose. Among them, 3142 number of documents are used for training and rest are used for testing. Total number of features is 24320 out of which 19886 are considered for training.

4.3 Reuters

Reuters is a widely used text mining dataset. It has 5485 training and 2189 testing documents classified into 8 classes, where all class documents are considered for evaluation. The total number of terms used is 17582 and among them 13531 are used for training.

4.4 WebKB

WebKB is a widespread text mining dataset in which the web pages are collected from four different college websites. It has 2803 training and 1396 testing documents classified into four classes, where all class documents are considered for evaluation. The total number of terms of all these documents is 7606 and from that 7522 terms are used for training.

4.5 Classic3

Classic3 is a widespread data mining dataset. It has 4257 training and 2838 test documents classified into 4 classes: *cacm, cisi, cran, med*, having 3204, 1460, 1400, and 1033 documents respectively. All the classes are considered in the evaluation. The total vocabulary contained in all documents is 21299 and from that 15971 terms are selected for training.

We compute the test accuracy, 100-Fold cross validation accuracy, precision, recall and F-Score

[2]https://www.dmoz.org/
[3]http://qwone.com/∼jason/20Newsgroups/
[4]http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/
[5]http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

for aiding our comparison of various kernel techniques. Tables 4 - 8 show the detail results of classification using SVMs on different kernels for various datasets (maximum results are marked as bold in each table).

4.6 Performance Evaluation

The following parameters are used to measure the performance of the classifier.

i. *Accuracy (acc)* is the ratio between the sum of true positive cases, *TP* (number of documents that are that are classified correctly) and true negative cases, *TN* (number of documents that are not classified correctly and are not retrieved by the approach) with the total number of documents, *N = TP + FP + TN + FN*. It can be represented as follows:

$$acc = \frac{(TP + TN)}{N}$$

where,
FP: number of documents that are not classified correctly and are retrieved by the approach and *FN*: number of documents that are classified correctly and are not retrieved by the approach.

ii. *Precision (pr)* is the fraction of the retrieved documents by the classifier that are relevant.

$$pr = \frac{(relevant_{documents}) \cap (retrieved_{documents})}{retrieved_{documents}}$$

iii. *Recall(re)* is the fraction of the relevant documents that are retrieved by the classifiers.

$$re = \frac{(relevant_{documents}) \cap (retrieved_{documents})}{relevant_{documents}}$$

iv. *F-measure (F)* is the harmonic mean of *pr* and *re*.

$$F = 2 * \left(\frac{pr * re}{pr + re}\right)$$

4.7 Discussion

From the results of all the tables, it is observed that positive definite kernels (RBF, Polynomial, and Euclidean) perform poorly in classifying the text documents; while conditionally positive definite kernels (Linear, Log-Cosine (L-cos), Log-Power (L-pow) and Negative Euclidean (N-Eue) performed significantly better performance. Although the Linear kernel based SVM performs reasonably well but this might be due to the fact that the rest of the positive definite kernels other than the linear kernel are exponential in nature. This highlights an inconsistency in the classification of text documents using positive definite kernel based SVMs.

The performance of the CPDs kernels is seen to be significantly more accurate, precise, and consistent than PDs. It is observe that both the log-power and negative Euclidean based SVMs are more effective in classifying the text documents. Both the log-power and negative Euclidean kernels deliver high precision which is significantly higher than their positive definite counterparts. It is important to note that the effective performance of the proposed modified cosine similarity measure i.e. the log-cosine distance kernel (L-cos) performs well compared to other CPDs on most of the datasets. SVMs with the negative log cosine kernel works well for text document classification and it is shown in Figure 1. Training score is 1 when the number of training examples = 1000. With increase in the number of training examples, training score decreases slightly to reach 0.99. Cross-validation accuracy increases at a decreasing rate with increase in the number of training examples and it reaches to more than 90%. Similarly, SVMs with log power kernel works well for text classification. As it can be inferred from the Figure 2 that the accuracy on training set remains 1 regardless of the number of training examples. Cross-validation accuracy increases at a decreasing rate with increase in the number of training examples and it reaches to 91%.

5 Conclusion

In this paper, we assess the effectiveness of classifying text documents using support vector machines on numerous kernel functions. Experimental results on five most popular machine learning datasets show that conditionally positive definite kernels perform more consistently and accurately than positive definite kernels. We also observed that PDs that are exponential in nature perform poorly in classifying the text documents, while the non-exponential linear kernel performs reasonably well. It is also proved that the proposed modified cosine distance based log kernel (L-cos) is indeed conditionally positive definite and generates the best results in comparison to all other existing kernels. A new feature selection technique

Table 4: SVM Classification on 20-NG dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	37.23	40.32	28.45	25.19	26.72
Poly	35.34	35.27	30.27	25.65	27.76
Eucl	32.03	35.18	22.15	22.06	22.10
Linear	80.62	84.05	80.27	79.72	79.99
L-cos	**85.06**	**87.83**	**83.81**	**82.81**	**83.30**
L-Pow	83.55	86.45	83.72	82.53	83.12
N-Euc	83.73	86.78	83.59	82.74	83.16

Table 5: SVM Classification on DMOZ dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	40.73	48.35	53.22	44.56	48.52
Poly	38.43	45.24	30.71	35.35	32.86
Eucl	51.72	50.34	42.65	39.30	40.90
Linear	82.56	79.23	80.76	77.27	78.97
L-cos	**84.38**	**85.81**	**81.84**	80.88	81.35
L-Pow	79.98	81.48	80.70	**83.35**	**82.00**
N-Euc	81.71	82.47	80.52	78.27	79.37

Table 6: SVM Classification on Reuters dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	32.12	35.47	28.78	35.87	31.93
Poly	35.53	35.28	30.87	31.98	31.41
Eucl	22.56	38.65	42.71	38.54	40.51
Linear	79.46	80.38	76.83	74.67	75.73
L-cos	**84.68**	86.51	**81.67**	**80.45**	**81.05**
L-Pow	81.35	86.43	77.67	78.93	78.29
N-Euc	80.57	**86.76**	80.54	78.45	79.48

Table 7: SVM Classification on Classic3 dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	37.65	35.72	28.36	29.30	28.82
Poly	47.33	48.22	30.30	35.87	32.85
Eucl	32.06	35.13	22.15	20.08	21.06
Linear	81.63	83.98	79.24	79.87	79.55
L-cos	**88.65**	83.88	**81.88**	**83.47**	**82.66**
L-Pow	82.66	82.47	76.34	74.58	75.44
N-Euc	80.75	**84.79**	75.55	77.64	76.58

is proposed which increases the performance of the classification results. To extend the work, it is needed to prove and adopt other conditionally positive definite functions as kernels for text documents classification. Further work can include the clustering of text documents on different datasets and observe the performance of these kernels using SVM.

References

[Aggarwal and Zhai2012] Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

[Bezdek et al.1984] James C Bezdek, Robert Ehrlich, and William Full. 1984. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203.

Table 8: SVM Classification on WebKB dataset

Kernel	Accuracy	100FAcc	Precision	Recall	F-score
RBF	35.25	35.27	21.26	25.87	23.34
Poly	34.63	29.45	22.76	19.45	20.97
Eucl	32.05	31.14	22.67	24.32	23.46
Linear	77.26	75.22	71.73	70.98	71.35
L-cos	83.23	**86.18**	**81.53**	**83.74**	**82.62**
L-Pow	**83.27**	80.34	70.34	71.65	70.98
N-Euc	81.74	82.67	72.34	74.56	73.43

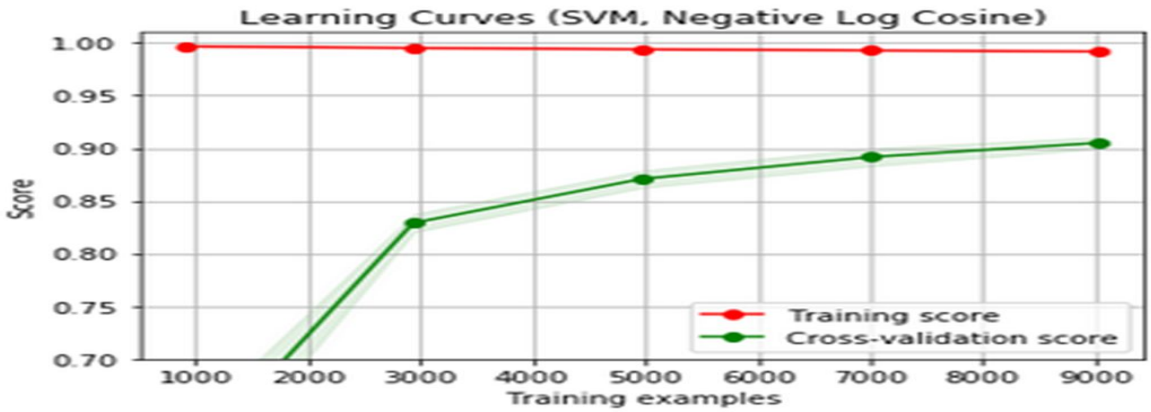

Figure 1: Training examples vs Accuracy (N-Log Cosine kernel)

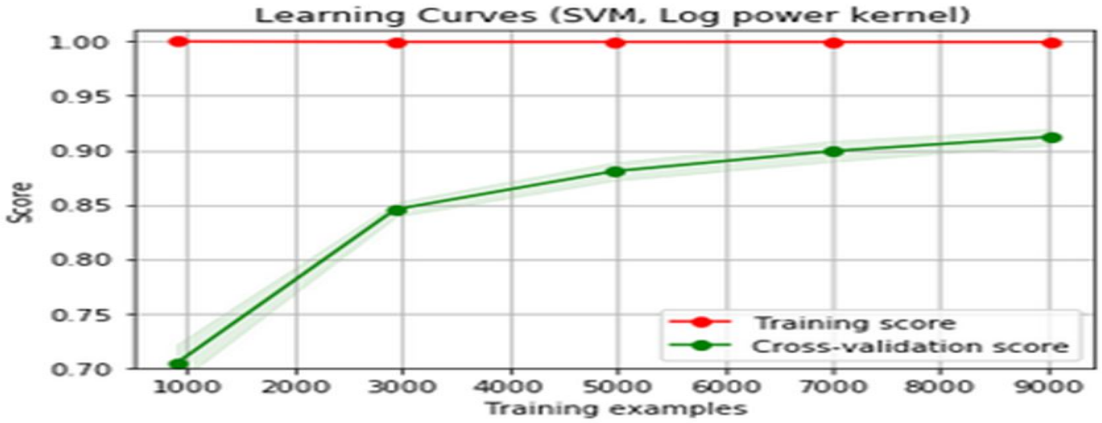

Figure 2: Training examples vs Accuracy (Log power kernel)

[Boughorbel et al.2005] Sabri Boughorbel, J-P Tarel, and Nozha Boujemaa. 2005. Conditionally positive definite kernels for svm based image recognition. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 113–116. IEEE.

[Cortes and Vapnik1995] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

[Cowling1983] Michael G Cowling. 1983. Harmonic analysis on semigroups. *Annals of Mathematics*, pages 267–283.

[Hamsici and Martinez2009] Onur C Hamsici and Aleix M Martinez. 2009. Rotation invariant kernels and their application to shape analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(11):1985–1999.

[Hong et al.2016] Haoyuan Hong, Biswajeet Pradhan, Dieu Tien Bui, Chong Xu, Ahmed M Youssef, and Wei Chen. 2016. Comparison of four kernel functions used in support vector machines for landslide susceptibility mapping: a case study at suichuan area (china). *Geomatics, Natural Hazards and Risk*, pages 1–26.

[Joachims1998] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142.

[Kira and Rendell1992] Kenji Kira and Larry A Rendell. 1992. The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134.

[Kohavi and John1997] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324.

[Lal et al.2006] Thomas Navin Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. 2006. Embedded methods. In *Feature extraction*, pages 137–165. Springer.

[Lodhi et al.2002] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

[MacQueen and others1967] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Maji et al.2013] Subhransu Maji, Alexander C Berg, and Jitendra Malik. 2013. Efficient classification for additive kernel svms. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):66–77.

[Ponte and Melko2017] Pedro Ponte and Roger G Melko. 2017. Kernel methods for interpretable machine learning of order parameters. *arXiv preprint arXiv:1704.05848*.

[Qi and Davison2009] Xiaoguang Qi and Brian D Davison. 2009. Web page classification: Features and algorithms. *ACM computing surveys (CSUR)*, 41(2):12.

[Qiu et al.2011] Xipeng Qiu, Xuanjing Huang, Zhao Liu, and Jinlong Zhou. 2011. Hierarchical text classification with latent concepts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 598–602. Association for Computational Linguistics.

[Roul and Rai2016] Rajendra Kumar Roul and Pranav Rai. 2016. A new feature selection technique combined with elm feature space for text classification. In *13th International Conference on Natural Language Processing*, pages 285–292.

[Roul and Sahoo2017] Rajendra Kumar Roul and Jajati Keshari Sahoo. 2017. Classification of research articles hierarchically: A new technique. In *Computational Intelligence in Data Mining*, pages 347–361. Springer.

[Roul et al.2016] Rajendra Kumar Roul, Aditya Bhalla, and Abhishek Srivastava. 2016. Commonality-rarity score computation: A novel feature selection technique using extended feature space of elm for text classification. In *Proceedings of the 8th annual meeting of the Forum on Information Retrieval Evaluation*, pages 37–41. ACM.

[Roul et al.2017] Rajendra Kumar Roul, Shubham Rohan Asthana, and Gaurav Kumar. 2017. Study on suitability and importance of multilayer extreme learning machine for classification of text data. *Soft Computing*, 21(15):4239–4256.

[Scholkopf2001] Bernhard Scholkopf. 2001. The kernel trick for distances. *Advances in neural information processing systems*, pages 301–307.

[Sebastiani2002] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

[Sparck Jones1972] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

[Tong and Koller2001] Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66.

[Zhang et al.2008] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. 2008. Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, 21(8):879–886.