Text Classification using Support Vector Machine

Indian Institute of
Information Technology, Allahabad

Under supervision
of
Dr. K P Singh

Motivation

- Document Classification can be automated using machine learning. Here, we categorize a text/document in one of the multiple predefined classes.
- By assigning categories to various documents (like customer reviews, emails, articles) we can use it in various application like spam detection, sentiment analysis, News categorisation, genre classification, etc.
- Our model uses supervised Machine Learning model to classify the documents into different categories.

Objective

- Applying various Natural Language Processing techniques for feature extraction from dataset.
- Applying SVD(Singular Value Decomposition) for feature reduction.
- To implement various kernel functions of SVM(Support Vector Machine) for classification:
  - $\rightarrow$ **linear kernel**
  - $\rightarrow$ **rbf (Gaussian) kernel**
  - $\rightarrow$ **Polynomial kernel**
  - $\rightarrow$ **Sigmoid kernel**
- Analyze the accuracy and training time for each mentioned kernel function of SVM by varying the parameters of SVC (Support Vector Classifier).

Literature Survey

S.no.	Paper Title	Year	Journal/Conference	Abstract	Author
1.	Inductive learning algorithms and representations for text categorization [1]	1998	7th International Conference on Information and knowledge management1998-11-01	In this paper five different algorithms for text classification have been compared like find similar, decision trees, naive bayes,bayes nets and SVM.The best result were found using SVM . So it guided us to choose SVM for classification.	Susan Dumais,Mehran Sahami,John Platt,David Heckerman
2	A Statistical Learning Model of Text Classification for-Support Vector Machines [3]	2001	24th annual international ACM SIGIR conference on Research and development in information retrieval2001-09-01	In this paper the statistical properties of text-classification tasks is connecccted with generalizaton performance of SVM. It explained why and when SVMs perform well for text classification.	Thorsten Joachims
3.	High-performing feature selection for text classification [4]	2002	11th International Conference on Information and knowledge management2002-11-04	This paper mainly focuses on the importance of feature selection and feature reduction in text classification. Here various techniques have been discussed on different machine learning algorithms for feature reduction.In this paper CHI_MAX and IG methods have been combined for giving weights. For reducing redundancy μ co-occurence method is used here.	Monica Rogati,Yiming Yang

Literature Survey (contd.)

4.	Transductive Inference for Text Classification using Support Vector Machines [2]	1999	16th International Conference on Machine Learning27-06-1999	In this paper the concept of Transductive SVM (TSVM) is introduced. It explains us the limitations of normal SVM in some cases where TSVM are better than SVM. The experiments here tells us about significant improvement of TSVM over inductive methods. TSVMs work very well in cases where there is smaller training dataset than the test set.	Thorsten Joachims
5.	An Optimal SVM-Based Text Classification Algorithm [5]	2006	International Conference on Machine Learning and Cybernetics13/09/2006	This paper describes new algorithms for feature selection which highly optimize the efficiency of classification. The new algorithm applies entropy weighing scheme for feature selection in a newer way. Also optimal parameter settings have been used to get better results.	Zi-Qiang Wang, Xia Sun, De-Xian Zhang, Xin Li

Methodology

1. **Getting the dataset:**
   The Reuters 21578 dataset is loaded and then sent for parsing to get it into a usable format.
2. **Parsing:**
   In this step we convert the dataset into a usable format which can be classified for our experiment. This conversion process is known as Parsing. Here we have made SGML parser class which overrides HTMLParser.
3. **Stop Words Removal:**
   Stop words are set of commonly used words in any language. It is important for us to filter these words in order to focus on more important words. For example : a, an, is, it, that, the, with, from, has, were, was, its, of, be, will, with, and, etc.
4. **Stemming:**
   It is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. [inflect (dictionary meaning) :- change the form of (a word) to express a particular grammatical function or attribute]
   Example :- (i) banks and banking become bank
   (ii)investing and invested become invest

Methodology (contd.)

5. **Vectorisation:**
   - It is used to convert raw text into a numerical data representation which can be used for classification.
   - Firstly we create list of tokens and normalise them using Bag of Words approach.
   - So entire dataset can be represented as a large matrix, each row representing one of the documents and each column representing token occurrence in that document. This is the process of vectorisation.
6. **tf-idf(Term-Frequency Inverse Document-Frequency):**
   - It gives us better weighting scheme of the tokens used for classifying the documents.
   - We get a high TF-IDF value if its frequency is high in that document and low frequency in collection of all documents

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

where:
$tf_{i,j}$ = number of occurences of i in j
df_i = number of documents containing i
N = total number of documents

Methodology (contd.)

7. **Singular Value Decomposition (SVD):**
   - We have used SVD here to reduce the number of features (Feature Reduction).
   - SVD achieves this task by creating new features which are linear combination of the existing ones.
   - Singular value decomposition reduces a matrix of R rank to a matrix of K rank.
   - Let A be a rank R matrix with m rows and n columns. SVD tells us :

$$\boxed{A = \mathbf{U}\Sigma V^T}$$ where :

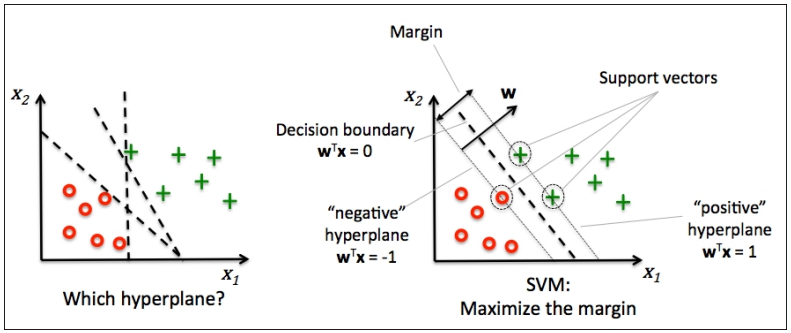
U $\rightarrow$is a square orthonormal m x r matrix. Its columns are the left singular vectors.
$\Sigma\rightarrow$is a diagonal r x r matrix with r positive values starting from the top left. These are singular values.
$V^T \rightarrow$is a square orthonormal r x n matrix. The rows are the right singular vectors.
In our context,
U$\rightarrow$no. of documents * no. of new features
$\Sigma \rightarrow$strength of new features in increasing order (positive)
$V^T\rightarrow$old features * new features.

Support Vector Machine(SVM)

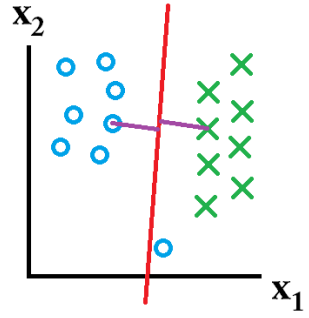
- Supervised machine learning model.
- Can be used for both classification (SVC) and regression (SVR).

SVM as a Classifier (SVC):

- Discriminative and non-probabilistic classifier.
- It classifies the different groups by finding the decision boundary that best separates the groups based on their known categories.
- This best decision boundary is the one that maximizes the margins between any two groups. **(Maximum Margin Classifier).**



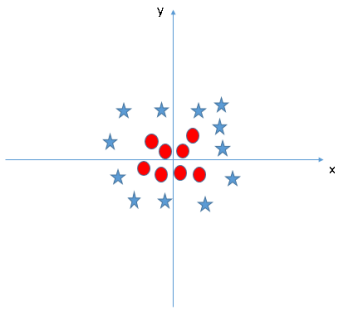
Maximum Margin Classifier and Support Vectors in SVM



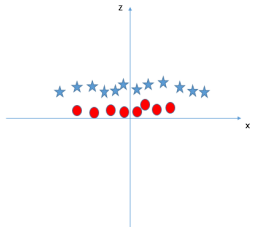
Outlier ignored in general SVM.

Support Vector Machine(SVM) (contd.)

- Linearly Separable Dataset $\rightarrow$ Simple
- Non-Linearly Separable Dataset $\rightarrow$ Not so simple (kernel trick used)



Non-linearly separable data



Mapping into higher dimension (adding z-axis component) $Z = x^2 + y^2$

- SVM does not need the actual vectors to work on it, it can do it only with the dot products between them. This dot product is called a kernel function.

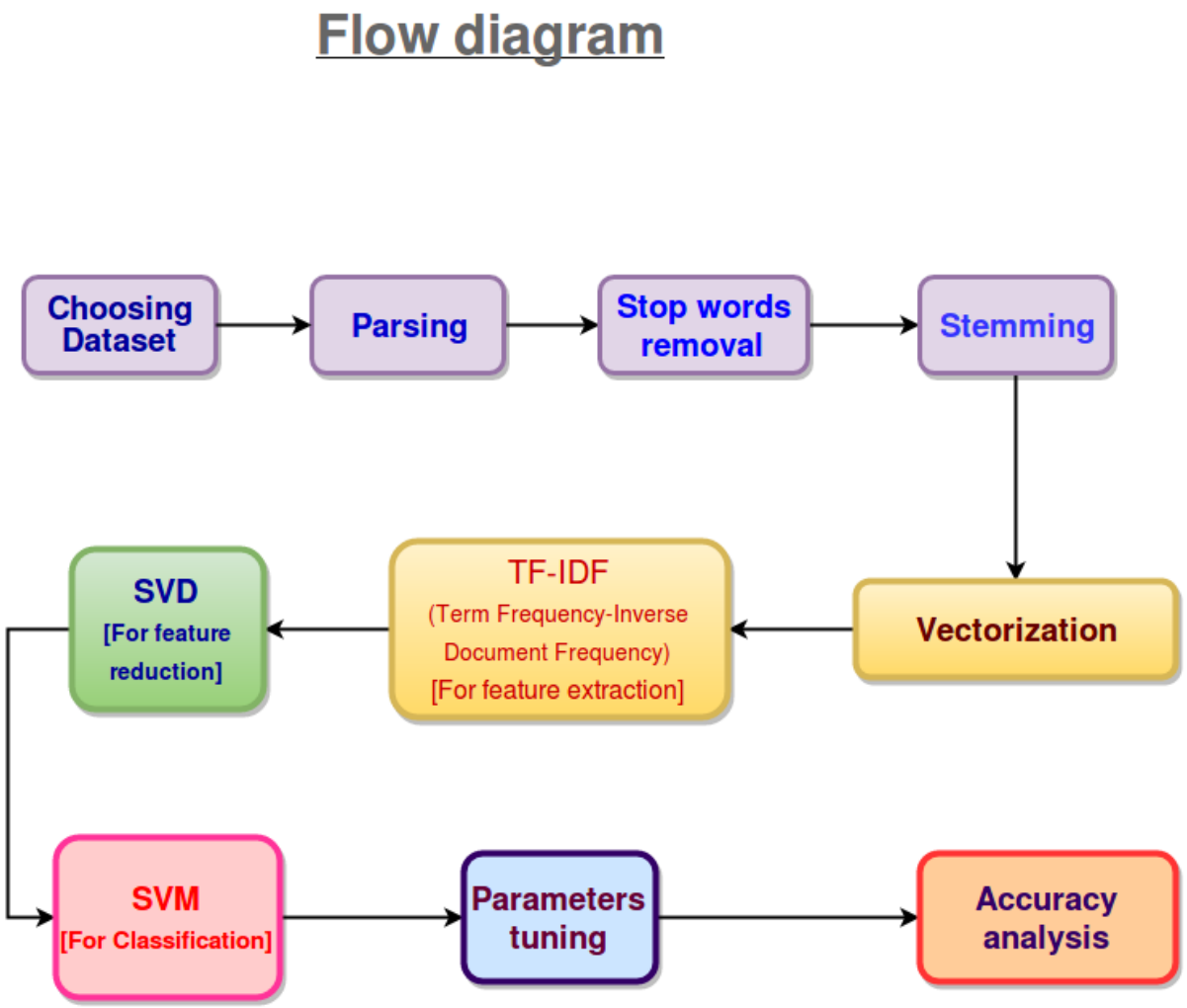
Kernel Functions

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma |\mathbf{X}_i - \mathbf{X}_j|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

where $K(\mathbf{X}_i, \mathbf{X}_j) = \phi(\mathbf{X}_i) \cdot \phi(\mathbf{X}_j)$

that is, the kernel function, represents a dot product of input data points mapped into the higher dimensional feature space by transformation ϕ

Methodology Flow Diagram



Experiments & Observations

Experiments and Observations

Change in **Accuracy** and **Training time** with change in SVC parameters **C** and **gamma**

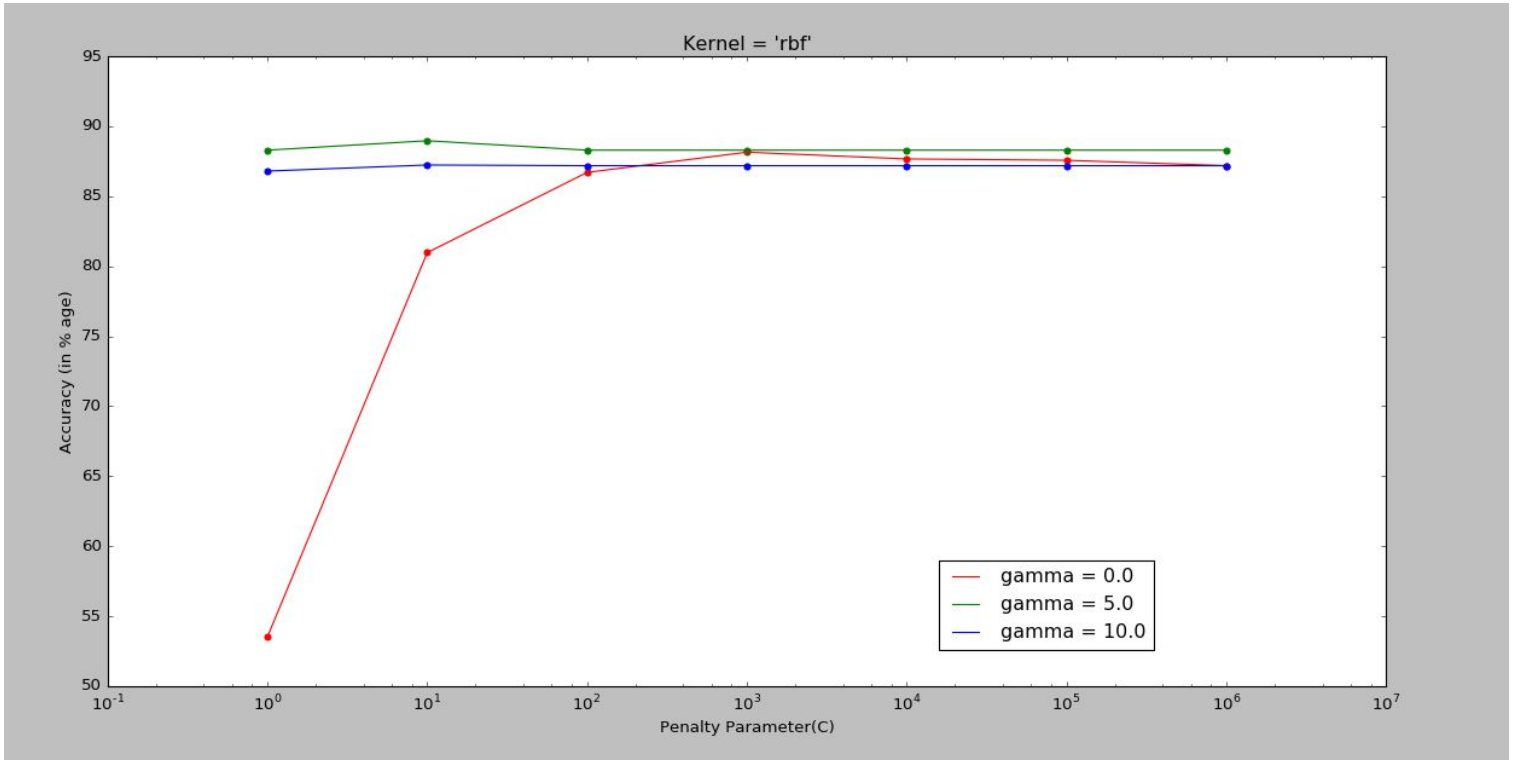
S.No.	Penalty Parameter (C)	Kernel Coef-ficient (gamma)	Training Time (in seconds)	Accuracy (% age)
1.1	1	0	14.7179185738	53.5163776493
1.2	1	5	20.2596582446	88.2947976879
1.3	1	10	49.7798475756	86.8015414258
2.1	10	0	10.9192837309	80.9730250482
2.2	10	5	19.5810598891	88.9691714836
2.3	10	10	50.6595395278	87.2350674374
3.1	100	0	7.2436635578	86.7052023121
3.2	100	5	19.2433398237	88.2947976879
3.3	100	10	50.6608579851	87.1868978805
4.1	1000	0	6.0687723209	88.1502890173
4.2	1000	5	19.1799743322	88.2947976879
4.3	1000	10	50.5428827899	87.1868978805
5.1	10000	0	6.2777937673	87.6685934489
5.2	10000	5	19.1768832492	88.2947976879
5.3	10000	10	50.9507252798	87.1868978805
6.1	100000	0	7.1739251665	87.5722543353
6.2	100000	5	19.1878773779	88.2947976879
6.3	100000	10	50.5921981372	87.1868978805
7.1	1000000	0	9.7328505405	87.1868978805
7.2	1000000	5	19.2118927153	88.2947976879
7.3	1000000	10	50.5413184316	87.1868978805

Kernel Function :
Radial Basis Function (rbf)

Using different values for the **Penalty Parameter (C)** and **Kernel coefficient (gamma)**, the best results we found from our experiments:

- C = 10.0
- gamma = 5.0
- Training Time taken = 19.581059889092217 seconds
- Accuracy on test dataset = 88.969171483622356 %

Experiments and Observations (contd.)



Accuracy vs **Penalty Parameter (C)** (for **rbf kernel**)

Standard deviation = 12.660401(for gamma = 0.0),
0.254889(for gamma = 5.0),
0.149765(for gamma = 10.0)

Experiments and Observations (contd.)

Change in Accuracy and Training time with change in SVC parameter C

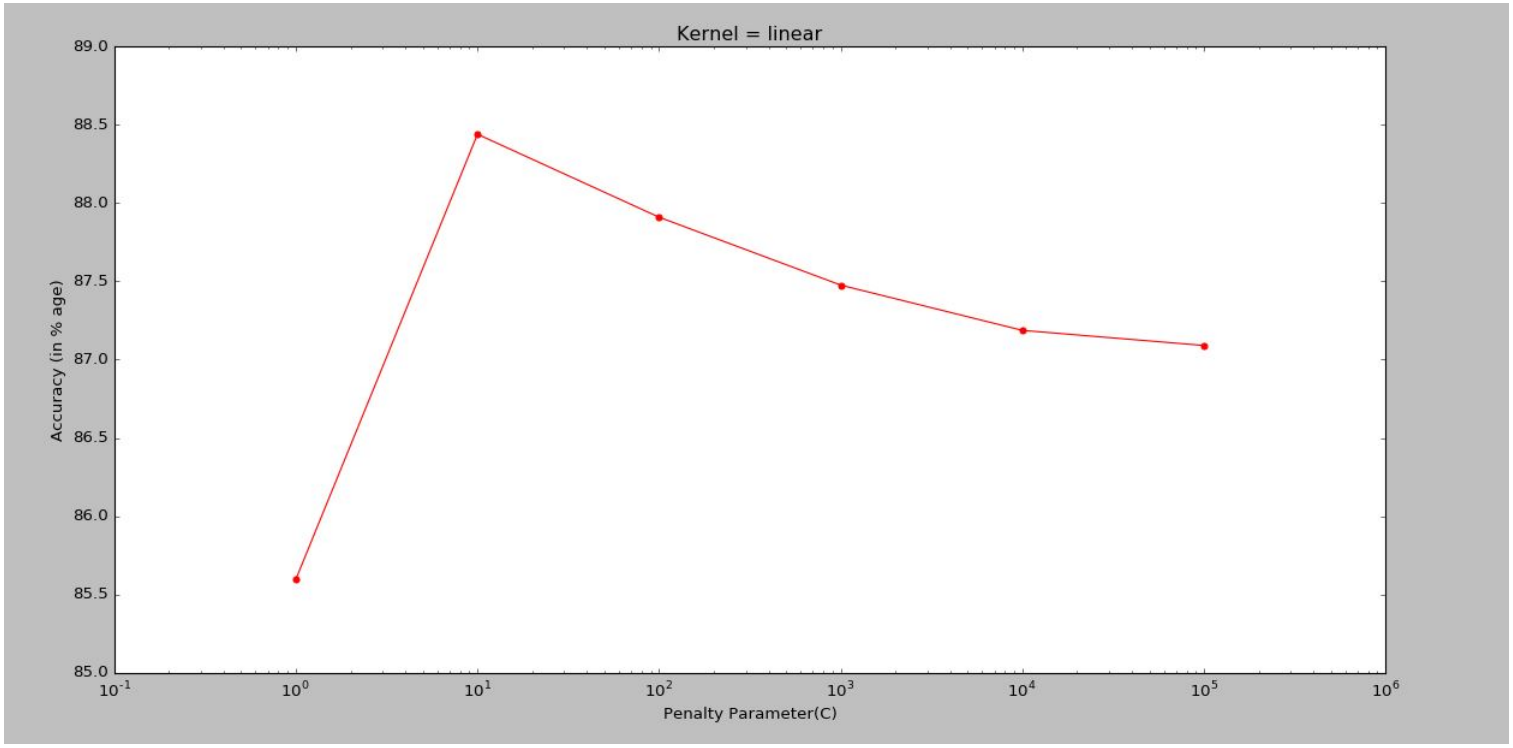
S.No.	Kernel	Penalty Parameter (C)	Training Time (in seconds)	Accuracy (% age)
1	Linear	1	5.7447666912	85.5973025048
2	Linear	10	4.4802451655	88.4393063584
3	Linear	100	4.4675985817	87.9094412331
4	Linear	1000	5.3948755933	87.4759152216
5	Linear	10000	10.4627913232	87.1868978805
6	Linear	100000	101.2622459789	87.0905587669

Kernel Function :
Linear

Using different values for the **Penalty Parameter (C)** and **Kernel coefficient (gamma)**, the best results we found from our experiments:

- C = 10.0
- Training Time taken = 4.48024516547855 seconds
- Accuracy on test dataset = 88.439306358381498 %

Experiments and Observations (contd.)



Accuracy vs **Penalty Parameter (C)** (for **linear kernel** with **gamma** = 0.0, 5.0 and 10.0))

Standard deviation = 0.964835

Experiments and Observations (contd.)

Average Accuracy and **Average Training Time** with different random splits of Training and Test

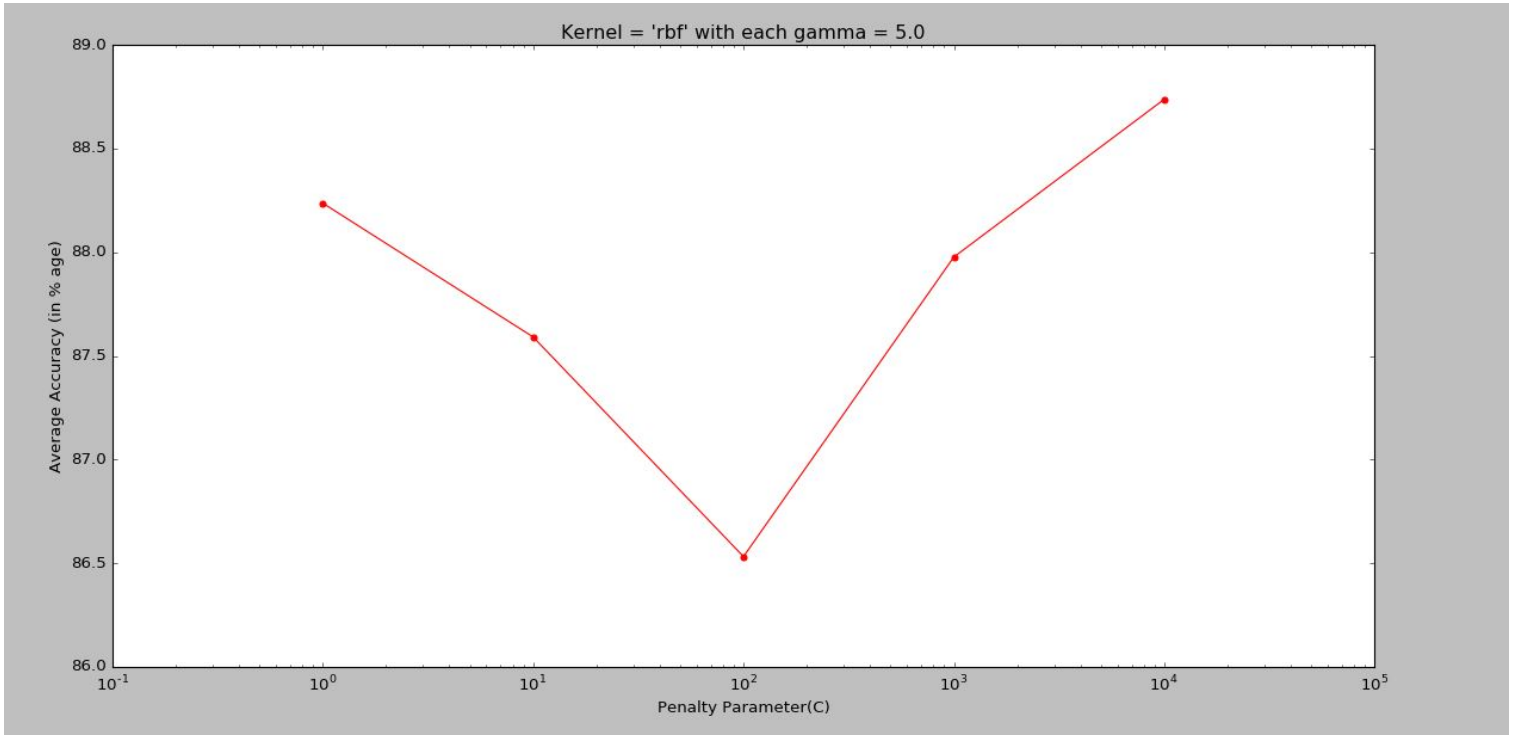
S.No.	Penalty Parameter (C)	Kernel Coefficient (gamma)	Average Training Time (in seconds)	Average Accuracy (% age)	Minimum Accuracy (% age)	Maximum Accuracy (% age)
1	10	5	19.5975905119	87.591522158	86.8497109827	88.1984585742
2	1	5	19.8280292122	88.2369942197	88.0539499037	88.3429672447
3	100	5	19.1036437403	86.5317919075	85.9344894027	86.7052023121
4	1000	5	18.7571186214	87.9768786127	87.1868978805	88.1984585742
5	10000	5	20.2823468282	88.7379576108	87.8612716763	89.2581888247

Kernel Function :
Radial Basis Function (rbf)

Using different random splits of training and test datasets for fixed value of SVC parameters, the best results we found are :

- C = 10000.0
- gamma = 5.0
- Average Training Time taken = 20.28234682823986
- Average Accuracy on test dataset = 88.737957610789986 %

Experiments and Observations (contd.)



Accuracy vs **Penalty Parameter (C)** (for **rbf kernel** with different splits of training and test data)

Standard deviation = 0.829563

Experiments and Observations (contd.)

Average Accuracy and **Average Training time** with different random splits of training and test

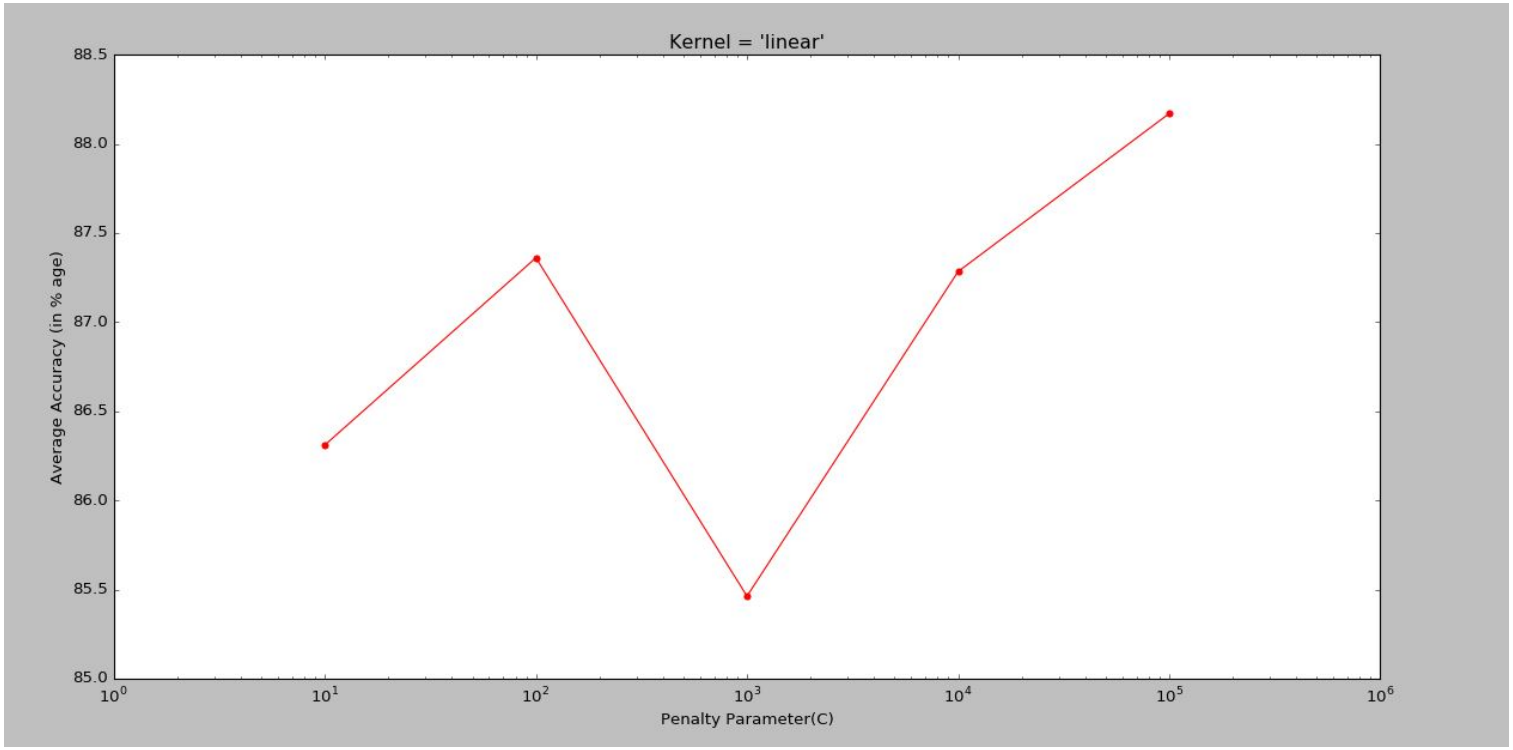
S.No.	Penalty Parameter (C)	Average Training Time (in seconds)	Average Accuracy (% age)	Minimum Accuracy (% age)	Maximum Accuracy (% age)
1	10	26.4541190135	86.3102119461	85.3082851638	87.6685934489
2	100	19.0949781875	87.3603082852	86.6570327553	88.8728323699
3	1000	26.5995590032	85.4624277457	84.4894026975	86.8978805395
4	10000	19.1290173923	87.2832369942	86.4161849711	88.5356454721
5	100000	22.8623470763	88.1695568401	87.4759152216	89.0655105973

Kernel Function :
Linear

Using different random splits of training and test datasets for fixed value of svc parameters the best results we found are:

- C = 100000.0
- Average Training Time taken = 22.862347076279367 seconds
- Average Accuracy on test dataset = 88.169556840077068 %

Experiments and Observations (contd.)



Accuracy vs **Penalty Parameter (C)** (for **linear kernel** with different splits of training and test data)

Standard deviation = 1.046843

Generic Text Classifier

After training and testing on "Reuters Dataset" we created Generic Text Classification model.
We tested our text classification model on various small datasets.

The average accuracy obtained for various datasets:-

1. Amazon review = 75.74% (data size = 1000 , kernel = linear)
2. IMDb review = 73.83% (data size = 1000 , kernel = linear)
3. Yelp review = 77.03% (data size = 1000 , kernel = linear)
4. Spam Detection = 96.34% (best we obtained till now , data size = 5574 , kernel = linear)
5. News Headline = 90.71% (data size = 5977 , kernel = rbf)

Amazon Online Review Binary-Classifier

For amazon review dataset we trained it and tested it on the splitted portion,
Once this part was over, we modified our model and made it online ,

1. now it takes review from user.
2. extracts and reduces the features complete dataset (original+user input) using SVD.
3. training is done only for our previous dataset.
4. then the review is accordingly classified
5. the user entered review is added to our dataset after classification

This added data helps us to improve accuracy if similar input is given again

Future Work

- Till now we have trained and tested our model on the same dataset. So there is a high chance of getting good accuracy.
- Our future challenge would be to train our model on one dataset and using its extracted features, try to classify(test) different but related dataset.
- As suggested by our instructor, we can solve this problem through Transfer Learning.

Transfer Learning

- Transfer learning is an emerging field in Machine Learning.
- It involves using knowledge extracted while solving one problem to apply it to a related but different problem.
- For instance, we could train our model on X news agency's(say FOX News) dataset and test on Y news agency's(say ABC news) dataset and analyse the features which lead to anomalous results.
- We could then adjust our model to these features, in order to develop a generic model which gives accurate results for most of the datasets in this domain(News).
- For transfer learning to work effectively we need to recognise and include those features in main features list which for effective knowledge extension.

Challenges

- During concluding part of our project,we worked on applying transfer learning in our model. We trained our model on News Articles Data(obtained from web scraping) dataset and tried to test it on another dataset which contains News Headlines only but we faced various challenges.
- For instance,when we trained our model with a larger dataset with 5000 features and test on a smaller dataset (say with 100 features,) our objective was to extract features from the bigger dataset which are common in both the datasets.Finally ,scale features of both the datasets on a common scale.
- We tried to use SVD for feature reduction but it didn't yield required results as applying SVD individually gives different scales for the two sets of features. The basic requirement for a effective transfer learning is that no. of features in training dataset should be equivalent to features in testing dataset.
- We look forward to overcoming this problem and work on new approach to implement transfer learning.

Softwares Used

IDE used :

- Spyder 3 (Scientific PYthon Development EnviRonment)

Language used :

- Python 3.5

Libraries Used :

- **pandas (Python Data Analysis Library)** for inputting the Dataset (.tsv file).
- The module **pyplot** of matplotlib library is used for graph plotting re library for the usage of Regular Expressions for text cleaning.
- Stopwords list from the '**corpus**' package of **nltk** (Natural Language Processing Toolkit) data package.
- **PorterStemmer** algorithm from the package nltk.stem.porter for stemming of words.

Softwares Used (contd.)

- **CountVectorizer** class from sklearn.feature_extraction.text submodule for building the matrix (sparse) of word counts from text documents.
- **TfidfTransformer** class from sklearn.feature_extraction.text submodule to convert the count matrix (sparse) to a matrix of TF_IDF features.
- **TruncatedSVD** class from sklearn.decomposition module for dimensionality reduction of the feature space.
- **train_test_split** function from sklearn.cross_validation module for splitting the whole Dataset into random train and test sunsets.
- **SVC** class from **sklearn.svm** module for classification of the test set.

References

[1] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami.
Inductive learning algorithms and representations for text categorization.
In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 148–155, New York, NY, USA, 1998. ACM.

[2] Thorsten Joachims.
Transductive inference for text classification using support vector machines.

[3] Thorsten Joachims.
A statistical learning learning model of text classification for support vector machines.
In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 128–136, New York, NY, USA, 2001. ACM.

[4] Monica Rogati and Yiming Yang.
High-performing feature selection for text classification.
In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 659–661, New York, NY, USA, 2002. ACM.

[5] Zi-Qiang Wang, Xia Sun, De-Xian Zhang, and Xin Li.
An optimal svm-based text classification algorithm.
In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 1378–1381. IEEE, 2006.