MACHINE LEARNING ALGORITHMS IN WEB PAGE CLASSIFICATION

W. A. AWAD

Math.& Computer Science Dept., Faculty of Science, Port Said University, Egypt.
Scientific Research Group in Egypt (SRGE), http://www.egyptscience.net
E-Mail: waelak71@yahoo.com

ABSTRACT

In this paper we use machine learning algorithms like SVM, KNN and GIS to perform a behavior comparison on the web pages classifications problem, from the experiment we see in the SVM with small number of negative documents to build the centroids has the smallest storage requirement and the least on line test computation cost. But almost all GIS with different number of nearest neighbors have an even higher storage requirement and on line test computation cost than KNN. This suggests that some future work should be done to try to reduce the storage requirement and on list test cost of GIS.

KEYWORDS

Web Classifications, Machine Learning, LIBSVM, SVM, K-NN.

1. Introduction

Nowadays, the web pages is growing at an exponential rate and can cover almost any information needed. However, the huge amount of web pages makes it more and more difficult to effectively find the target information for a user. Generally two solutions exist, hierarchical browsing and keyword searching. However, these pages vary to a great extent in both the information content and quality. Moreover, the organization of these pages does not allow for easy search. So an efficient and accurate method for classifying this huge amount of data is very essential if the web pages is to be exploited to its full potential. This has been felt for a long time and many approaches have been tried to solve this problem. Many different Machine learning-based algorithms have been applied to the text classification task, including k-Nearest Neighbour (k-NN Algorithm) [2], Bayesian algorithm [7], Support Vector Machine (SVM) [5], Neural Networks [8], and decision trees [9].

2. Related Work

In past years, there have been extensive investigations and rapid progresses in automatically hierarchical classification. Basically, the models depend on the hierarchical structure of the dataset and the assignment of documents to nodes in the structure. Nevertheless, most of researches adopt a top-down model for classification. M.-Y. Kan and H. O. N. Thi. [9]; have summarized four structures used in text classification: virtual category tree, category tree, virtual directed acyclic category graph and directed acyclic category graph. The second and forth

structures are popularly used in many web directory services since the documents could be assigned to both internal and leaf categories. These two models used category-similarity measures and distance-based measures to describe the degree of falsely classification in judging the classification performance. F. Sebastiani [5]; found a boost of the accuracy for hierarchical models over flat models using a sequential Boolean decision rule and a multiplicative decision rules.

They claimed only few of the comparisons are required in their approaches because of the efficiency of sequential approach. They also have shown that SVM take a good performance for virtual category tree, but category tree is not considered in their work. E. Gabrilovich and S. Markovitch [6]; proposed an efficient optimization algorithm, which is based on incremental conditional gradient ascent in single-example sub-spaces spanned by the marginal dual variables. The classification model is a variant of the Maximum Margin Markov Network framework, which is equipped with an exponential family defined on the edges. This method solved the scaling problem to medium-sized datasets but whether it is fit for huge amount data set is unclear. P. N. Bennett and N. Nguyen[2]; used a refined evaluation, which turns the hierarchical SVM classifier into an approximate value of the Bayes optimal classifier with respect to a simple stochastic model for the labels. They announced an improvement on hierarchical SVM algorithm by replacing its top-down evaluation with a recursive bottom-up scheme. Loss function is used to measure the performance in classification filed in this paper. The aim of our work is to discuss the problem of imbalanced data and measurement of multi-label classification and find strategies to solve them. In the experiments, we compare the classification performance using both the standard measurements precision/recall and extended measurement loss value.

3. Machine Learning Algorithms

3.1. Support Vector Machines (SVM)

Support Vector Machine is an machine learning technique based on Statistical Learning theory. SVM has been proved to be very effective in dealing with high-dimensional feature spaces, the most challenging problem of other machine learning techniques due to the so-called curse of dimensionality. For simplicity, let's examine the basic idea of SVM in the linear-separable case (i.e., the training sample is separable by a hyper plane). Based on Structural Risk Minimization principle, SVM algorithm try to find a hyper plane such that the margin (i.e., the minimal distance of any training point to the hyper plane, Figure 1) is optimal. In order to find an optimal margin, quadratic optimization is used in which the basic computation is the dot product of two points in the input space.

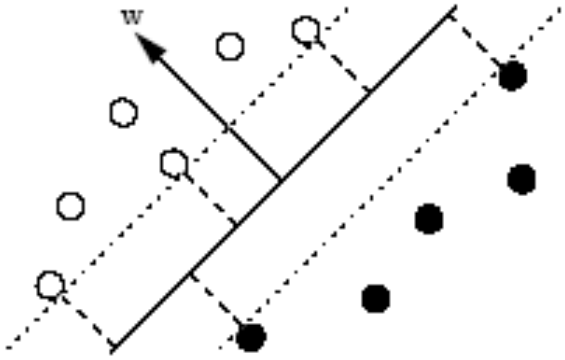

Figure 1: Linear Classifier and Margins

For nonlinear-separable case, SVM use kernel-based methods to map the input space to a so-called feature space. The basic idea of kernel methods is finding a map Φ from the input space which is nonlinear separable to a linear-separable feature space ; Figure 2. However, the problem with the feature space is that it is usually of very large or even infinite dimensions and thus computing dot product in this space is intractable. Fortunately, kernel methods overcomes this problem by finding maps such that computing dot products in feature spaces becomes computing kernel functions in input spaces

$$k(x,y) = \langle \Phi(x).\Phi(y) \rangle;$$

where k(x,y) is a kernel function in the input space and $\langle \Phi(x).\Phi(y) \rangle$ is a dot product in the feature space. Therefore, the dot product in feature spaces can be computed even if the map Φ is unknown. Some most widely used kernel functions are Gaussian RBF, Polynomial, Sigmoidal, and B-Splines.

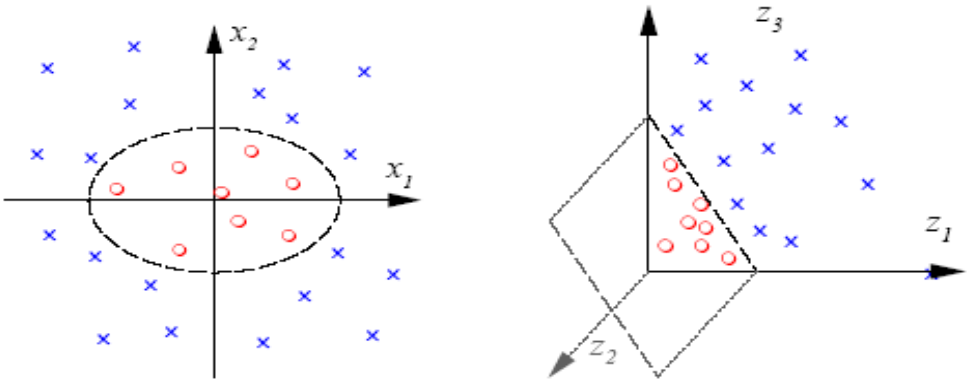

Figure 2: Example of mapping from input space to feature space.

3.2. K-Nearest Neighbour

In these methods, there is a family of so-called Memory Based Learning or instance-based learning algorithms like kNN and Rocchio. In these algorithms, in contrast to learning methods that construct a general, explicit description of the target function when training examples are provided. Those algorithms simply store the training examples as they are or transform the training examples into some other objects before store them. All those objects are called instances or cases. When each time a new query instance is encountered, its relationship to all the previous stored instances is calculated in order to assign a target function value for the new instance. This kind of algorithm typically has a low off line computation training cost but a high storage requirement and a high on line test computation cost.

In most instanced based learning methods, they use a common technique to represent a document. Each document is a feature vector of the form $(d_1,\ldots\ldots,d_m)$,here di is the ith feature of the document and m is the number of features. Typically, m will be the vocabulary size and each feature represents a specific word. Different weighting schemes to determine the weights have been introduced and most of them include binary weight, term frequency and combined with inverse document frequency. Sometimes these weights are normalized by divided by a factor of the factor norm. K-nearest neighbor (kNN) alogirthm is one of the instance-based learning algorithms. Expert Network [3]; is known as one of best text categorization classifier designed with this algorithm. In the training phase, each document is preprocessed to a vector as descript before and stored in the instances collection. In the online training phase, the similarity between query document an each instance is calculated as:

$$S(X,D_j)=\frac{\sum_{i=1}^{m} x_i d_{ij}}{\sqrt{\sum_{i=1}^{m} x_i^2}\sqrt{\sum_{i=1}^{m} d_{ij}^2}}$$

The score of documents belong to every corresponding category is accumulated and a threshold can be set to assign some categories to this new unknown document. Only the score of the k nearest neighbor documents of this new unknown document query is taken into account.

3.3. Generalized Instance Set (GIS) Algorithm:

So is there some method that can automatically adjust the granularity of the training instance set size and keep SVM and kNN's advantages as much as possible? Generalized Instance Set (GIS) [4]; is such an algorithm for this idea. GIS selectively substitutes appropriate positive and negative training documents in the training document pool to remove some noise in the training set. It automatically selects a representative positive instance and performances a generalization. Actually it does this generalization procedure in a supervised way. It brings out an evaluation metric to measure the representation ability of a generalized instance set and does the generalization steps in a greedy to increase the representation ability. The representative function Rep(G) for a generalized instance G is defined as follows:

$$\mathrm{Re}\, p(G) = \sum_{I^+ \in K} (k - rank(I^+))$$

The pseudo-code is as follows:

```
Input:  The training set T
        The category C
Procedure GIS(T,C)
 Let G and G_new be generalized instances.
 GS be generalized instance set, and GS is initialized to empty.
 Repeat
      Randomly select a positive instance as G.
      Rank instances in T according to the similarity
      Compute Rep(G)
      Repeat
          G_new=G
          G=Generalize(G_new,k nearest neighor)
          Rank instances in T according to the similarity
          Compute Rep(G)
       Until Rep(G)<Rep(G_new)
       Add G_new to GS
       Remove top k instances from T.
  Until no positive instances in T
  Return GS
```

In this algorithm, we use a SVM formula that is described in previous section for the generalization step.

4. Training Phase:

The SVM classifier of our application is implemented based on the LIBSVM library. The library realizes C-support vector classification (C-SVC), ν-support vector classification (ν-SVC), ν-support vector regression (ν-SVR), and incorporates many efficient features such as caching, sequential minimal optimization and performs well in moderate-sized problems (about tens of thousands of training data points). In our application, we only use C-SVC with RBF kernel function

$$K(x, y) = e^{-\gamma \|x-y\|^2}.$$

To train our SVM classifier we use a collection of 7566 Aljazeera News web documents that are already categorized in 16 broad categories by domain experts. After parsing and pre-processing, the total number of terms are 189815, meaning that each document is represented by a 189815-dimensional sparse vector – a large dimension that is considered to be extremely difficult in other machine learning techniques such as neural networks or decision trees. Aside from the training data set, another testing data set consisting of 490 web documents that are also already classified in the same 16 categories is used to evaluate the classifier. To decide the best

parameters (i.e., C and γ) for the classifier, a grid search (i.e., search over various pairs (C, γ)) is needed to be performed. However, since a full grid search is very time-consuming we fix the γ parameter as 1/k (k is the number of training data) and only try various values of C. A recommended range of values of C is 2^0, 2^1, 2^2, … , 2^{10} which is known good enough in practice. The classification accuracies obtained over training and testing data with various values of C are shown in Table 1 below.

Table 1: Classification Accuracy over Training and Testing Data for Various Values of C Parameter.

C	*Accuracy over training data*	*Accuracy over testing data*
1	59.91%	44.90%
2	69.53%	53.06%
4	78.09%	57.35%
8	85.47%	61.22%
16	90.92%	64.69%
32	94.14%	68.57%
64	96.52%	71.22%
128	97.82%	71.02%
256	98.23%	71.02%
512	98.49%	72.45%
1024	98.61%	72.45%

As we can see in Table 1, with C=1024 the classifier performs most accurately. Therefore, we choose C=1024 for our classifier.

5. Dataset:

We use a collection of 21578 Aljazeera News as the Dataset for the algorithms. It is a small corpus. But it is a widely used corpus in the text categorization task and it is easy to use this corpus to compare with the performance of other algorithms.

Dataset	Training documents	Test documents
Aljazeera News 21578	7769	3019

5.1 Experiment Results:

Algorithm	Precision	Recall	F1
SVM	0.781316	0.861111	0.819314
KNN	0.83814	0.855740	0.846849

GIS	0.845085	0.853060	0.849054

Here we use all negative documents as well as positive documents to build the centroid for every category in SVM; for kNN we choose k as 45; and for GIS we choose the number of nearest neighbors as 40.

5.2 Result Analysis:

5.2.1 Negative Documents in SVM:

There have been some arguments for the effectiveness of using negative documents in constructing the centroids for the GIS algorithm. Some people claimed that using a rather small number of negative is enough for GIS. Some experiments are done for this problem. In the following graph, the x-axis represents the number of negative documents used to build the centroids. The y-axis is the performance of GIS algorithm that is tried with 10-fold cross validation.

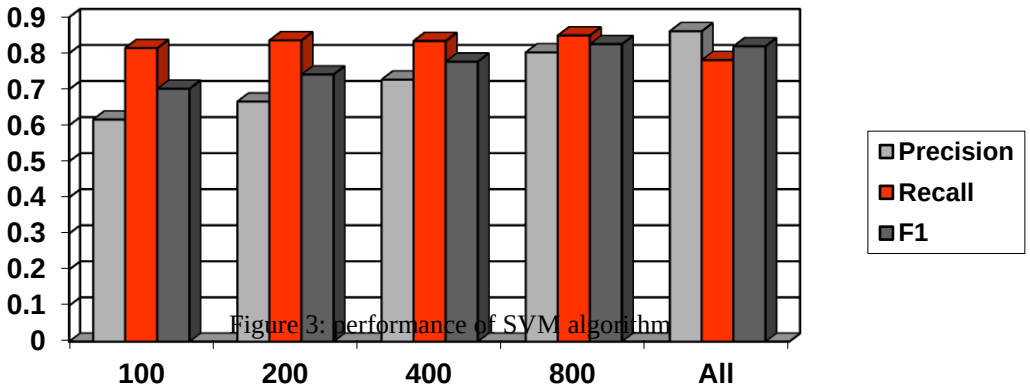

Figure 3: performance of SVM algorithm

We can see that the performance of SVM algorithm does has relationship with the number of negative documents used to build the centroid in the corpus. And it is interesting to see that precision seems to increase monotonically with the number of negative documents. It can be explained intuitively, as more and more negative documents are subtracted from the centroid, the non-relevant document will have a larger distance with the centroid so the precision will get higher. But the recall's trend is more complex.

5.2.2 The Number of K nearest neighbors in GIS

In the following graph, the x-axis represents the number of k nearest neighbors used to build the generalized instance, and y-axis represents the performance of GIS algorithms by 10-fold cross validations.

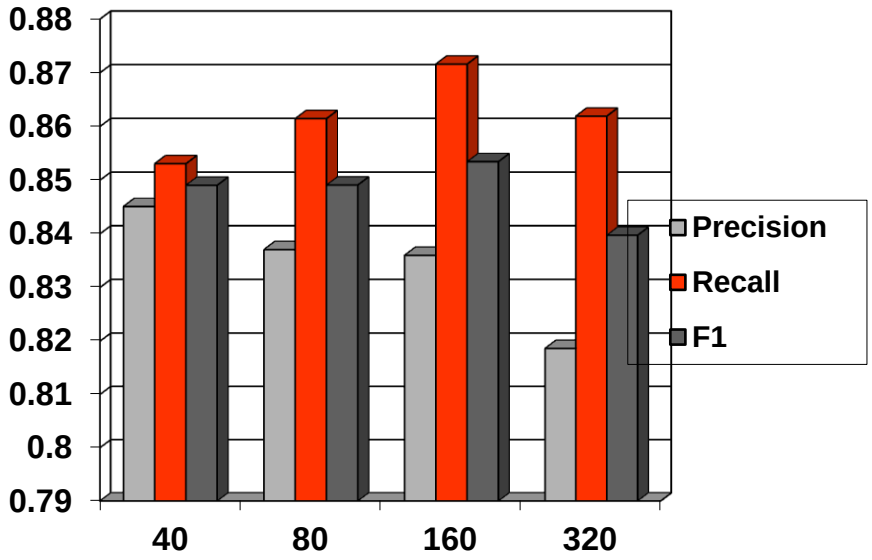

Figure 4: Performance of GIS Algorithms

We can see from the graph that there is a tradeoff between the number of nearest neighbors and the performance. Intuitively speaking, a too small number of nearest neighbors cannot give a very much generalization power to the centroid and cannot make the centroid insensitive to the noise too much. But too much number of the nearest neighbors will ignore the local detail information. So there should be such kind of trade off in the number of near neighbors' choice.

6. Conclusion:

From the graphs, we can see SVM with small number of negative documents to build the centroids has the smallest storage requirement and the least on line test computation cost. But almost all GIS with different number of nearest neighbors have a even higher storage requirement and on line test computation cost than KNN. This suggests that some future work should be done to try to reduce the storage requirement and on list test cost of GIS. But in reality, people often only keep most important (high weight) terms in each centroid instead of keeping everything. So in that case, the analysis does not make much sense. But if we cut off the number of terms, the performance will be influenced. There is also some tradeoff for this factor. And that is something that we should do more work to investigate. GIS is some idea that wants to find the true granularity for instances' set in an automatically and supervised way. There was some other work that tries to do the same work in unsupervised way [5]. Their results are different on two different Datasets. It seems that our experiment does not support the effectiveness of this idea. But there needs more work to explore this work.

7. References

[1] E. Baykan, M. Henzinger, L. Marian, and I. Weber. A comprehensive study of features and algorithms for url-based topic Classification. ACM Transactions on the Web, 5:15:1–15:29, July 2011.

[2] P. N. Bennett and N. Nguyen. Refined experts: improving Classification in large taxonomies. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 11–18. ACM, 2009

[3] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust Classification of rare queries using web knowledge. In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 231–238, New York, NY, July 2007. ACM Press

[4] C. Castillo and B. D. Davison. Adversarial web search. Foundations and Trends in Information Retrieval, 4(5):377–486, 2010.

[5] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1–47, March 2002.

[6] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In Proceedings of the 21st International Conference on Machine Learning, page 41, New York, NY, 2004. ACM Press.

[7] H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. In 18th International World Wide Web Conference, pages 201–201, April 2009.

[8] C.-C. Huang, S.-L. Chuang, and L.-F. Chien. Liveclassifier: Creating hierarchical text classifiers through web corpora. In Proceedings of the 13th International Conference on World Wide Web, pages 184–192, New York, NY, 2004. ACM Press

[9] M.-Y. Kan and H. O. N. Thi. Fast webpage Classification using URL features. In Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pages 325–326, New York, NY, 2005. ACM Press.

[10] H. Malik. Improving hierarchical svms by hierarchy flattening and lazy Classication. In Proceedings of Large-Scale Hierarchical Classification Workshop, 2010.

[11] X. Qi and B. D. Davison. Hierarchy evolution for improved Classification. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pages 2193–2196, October 2011.

[12] W. A. Awad, S. M. ELseuofi ; "Machine Learning Methods for Spam E-Mail Classification"; Int. J. of Computer Applications (IJCA); Vol. 16; No. 1; 2011.

Authors

W. A. Awad,
Mathematics & Computer Science Department,
Faculty of Science, Port Said University, Egypt,
E-Mail: Waelak71@yahoo.com