Active Learning SVM for Blogs recommendation

Xin Guan
Computer Science, George Mason University

Ⅰ. Introduction

In the DH Now website, they try to review a big amount of blogs and articles and find the worth-to-read one as Editor's Choice in their specific field, and recommend to readers. There are more than 3,000 articles need to be reviewed every month, while only around 20 articles would be selected as Editor's choice. The "DH now" opened on Sep, 2011. Till Dec, 2012, only 260 were accepted by editors and marked "Editor's choice". It would be very time consuming job, with low value, to review all of the articles. Our goal is to find out such blogs using data mining methods. The data we got are all coming from internet. The editors follow more than 150 websites and blogs, and download the blogs and news using RSS reader.

We consider this problem as a classification problem. There are only two classes, one is "accept", the other is "reject". The training data set was given by Joan, an editor of "DH Now" website, containing around 2,500 articles that published During July, 2012. And 235 the editor's choice articles that published from Nov 2011, to Sep 2012. Among these articles there would be some duplicated articles, and some wrong labeled articles, those worthy to read but missed by editors. Also the lengths of these articles are different from each other.

Active Learning is an idea that makes the classifier themselves to choose which instances' label they want to know. This kind of algorithm will be considered to use when the label is expensive or not easy to get, and we need to reduce the need of instances as few as possible. It also has better performance when the data is imbalanced, because the classifier will automatically choose the most usable instances from the unlabeled instances pool. In this article we will use Active Learning method to improve the performance of classifiers.

In this article, we will introduce the Active learning SVM algorithm to the classification problem, and we also show our work on pre-processing the article to matrix, the methods we choice to classify the blogs, and the performance compare with other classifiers.

Ⅱ. Method

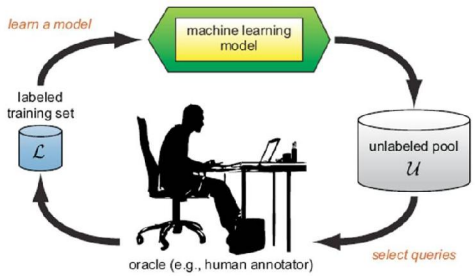
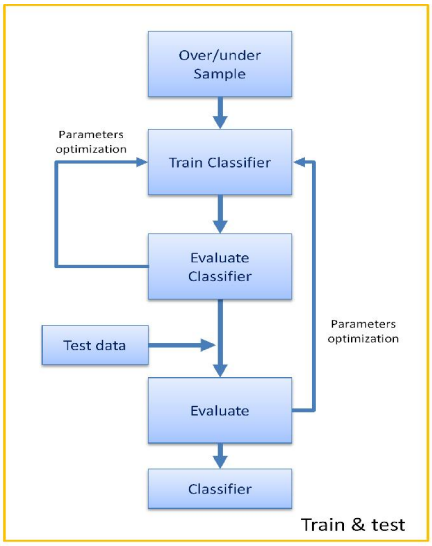


Figure 1. Pool-based Active Learning Cycle[3]

Active Learning is a subfield of machine learning and, more generally, artificial intelligence. The idea behind it is that if a machine learning algorithm is allowed to

select the data from the pool and get the labels of these data, it can achieve greater accuracy with few training labels. Active learning is well-motivated in many modern machine learning problems where data may be abundant but labels are scarce or expensive to obtain. One kind of Active learning is called the pool-based active learning. In the pool-based active learning cycle, a machine learning model will train with a small amount of instances. And then the machine leaning model will make a query based on a large unlabeled instances pool, and ask the oracle (e.g. human annotator) to give the label to that instance. And then use the new information to update the model itself.[3] Actually, every cycle it will output a model for classification, but usually, we will wait several cycles for the active learning model to improve its performance. The flow chart for normal classification is shown in Figure 2, and the flow chart for Active learning classification is shown in Figure 3. From the flow chart and the above statements we can figure out that the most important difference between the normal classifier and active learning classifier is how the training data was given. In normal classifiers, the training samples are usually given by random, and all of them will be used during the training step. Sometimes the researchers will use cross-validation to evaluate the model, but they still decide the training data randomly. While in active learning classifier doesn't. The active learning model will query several instances from a large unlabeled instances pool not based on randomly choose, but based on the query strategies. And these selected instances will be given the ground truth label and then join in the training data set for the classifier to update.



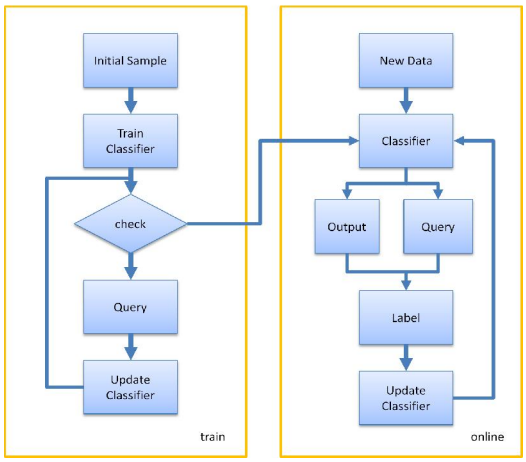


Figure 3. Flow chart for Active Learning classification

To use Active learning algorithm in the blog recommendation problem, we choose to implement Active Learning SVM as the classifier. For updating the classifier after each query, we just re-train the SVM with the given instances with label and add the new queried instances. For query strategy, we choose the instances that close to the hyper-plane and with high variance to the training data set. Figure 4 show the support vectors and help explain how the intuitive way of query works.

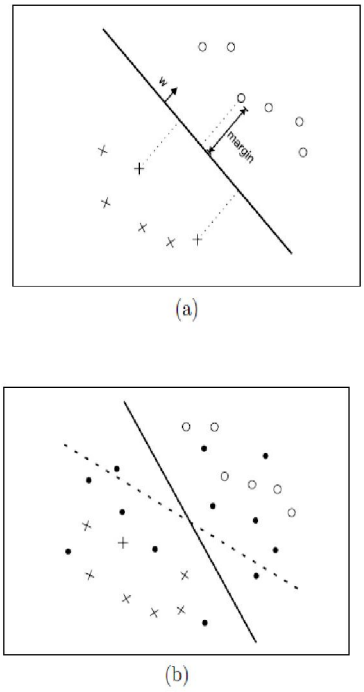


Figure 4. Support Vector Query Strategy

As shown in Figure 4. The solid circles represent unlabeled instances. The start and circle represent labeled instances for two classes. The solid line is the current hyper-plane based on labeled instances. And the dot line is the ground truth hyper-plane we want to get. To query instances from unlabeled data, an intuitive way is to select the instances that close to hyper-plane, because if these instances' labels are known, the hyper-plane will have more constrains and be limited to a certain space, called version space. *Simon Tong (2001)* had prove that selected these instances will fast reduce the size of version space, and then get the hyper-plane that very close to the expect hyper-plane. The distance is calculated by formula (1). And we will query the instances which have the minimum value of distance.

$$f(x_i) = \mathcal{W} \cdot \phi(x_i) = \sum_{j=1}^{N} \alpha_j y_j \mathcal{K}(x_i, x_j) \quad (1)$$

$$Query = min(|f(x_i)|) \quad (2)$$

King-Shy Goh et al(2004) supposed that the queried instances should contain some information that the current labeled data set does not have. So they proposed a query method called angle diversity, which besides considering the distance to the hyper-plane, also consider the queried instances' max cosine value to the already labeled instances. They assume such instances would be more helpful to the future data. They use the formula 3 to get the score of unlabeled instances, and query the lowest one.

$$\left|\cos\angle\left(\phi(x_i), \phi(x_j)\right)\right| = \frac{|\phi(x_i)\cdot\phi(x_j)|}{\|\phi(x_i)\|\|\phi(x_j)\|} = \frac{|\mathcal{K}(x_i,x_j)|}{\sqrt{\mathcal{K}(x_i,x_i)\mathcal{K}(x_j,x_j)}} \quad (3)$$

$$Query = \min(\lambda|f(x_i)| + (1-\lambda)\max_{x_j \in S} \frac{|\mathcal{K}(x_i,x_j)|}{\sqrt{\mathcal{K}(x_i,x_i)\mathcal{K}(x_j,x_j)}}) \quad (4)$$

Ⅲ. Implementation

First, pre-process the data set to matrix. The data set are downloaded by Google reader API, and it is a XML file. After the common pre-process for documents, including change to lowercase letter, delete stop words, transform to feature vector based on dictionary, we will get a matrix with very high dimension. Each dimension is represent a word in the dictionary, and the value for that dimension is the frequency of this word appeared in one document. Thanks to the Mallet project, we can easily do the above process using their open source library. The label for each document was manually obtained by compare the title with the already selected "Editor's choice" on the "DH Now" website.

There already are plenty of classifiers were introduced to data mining field. We choose some very famous and popular classifiers as our bench mark for the blog recommendation problem. They are Naïve Bayes, Decision Tree (C45), and Max entropy. What's more,

Support Vector Machine (SVM) was already shown its success in text classification, so we also choose SVM as one of the classifier for the performance bench mark. The library of Mallet contains Naïve Bayes, Decision Tree, and Max entropy, and for SVM we use libSVM for Java.

For Active Learning SVM implementation, according to the formula stated in previous part, we can have the pseudo- code for Active Learning SVM as following. After every cycle of the Active Learning SVM, we will get an output. And the output is the classifier we used.

Input:L, U, h, λ;
Output:L, f
Procedures:
SVM_train () /*SVM Training Algorithm
f()/*Classifier Learned by SVMs On L
K()/*SVM Kernel Function
Label();/* label the queried sample
Initialization:
Begin:
$S \leftarrow \emptyset$;
1. $f \leftarrow$ SVM_train (L);
2. *while*($|S| < h$)

$$x_s \leftarrow \arg\min_{x_j \in U} (\lambda |f(x_i)| + (1-\lambda) \max_{x_j \in L} \frac{|K(x_i, x_j)|}{\sqrt{K(x_i, x_i) K(x_j, x_j)}})$$

$S \leftarrow S \cup \{x_s\}$
3. *label*(S)
4. $L \leftarrow L \cup S$
Return f, L
End

Figure 5. Active Learning SVM pseudo-code[1]

We have 2736 instances training data in total, in which 235 are positive, and rest of them are negative. Because of the imbalance amount between positive instances and negative instances, we first under sample the training set to match each other. For the normal classifiers, we generate two data sets for classifier evaluating. For the first one, under sample the negative to match the positive instances, and keep the ratio of positive to negative 1:1. I get 460 instances, and call it data set 1. And second one, under sample negative to match the positive instances, and keep the ratio of positive to negative 1:2. I get 690 instances, and call it Data set 2. For the active learning, we split the data set to two parts, 2/3 of them are training, and 1/3 of them are testing, and keep the ratio of positive to negative the same in training set and testing set. The results of the experiment are shown in part 4.

Ⅳ. Results

First show the performance of the normal classifiers. Using 10 folder cross validation to evaluate the classifier we get the results as following. Table1 shows the performance on Data set 1, (Pos: Neg= 1:1). Table 2 shows the performance on Data set 2 (Pos:Neg= 1:2).

Table 1. Measurement on Data set 1(Pos: Neg= 230:230)

	Accuracy	Recall	Precision
Naïve Bayes	0.8588(0.0546)	0.9611(0.0321)	0.8038(0.0741)
Decision Tree(C45)	0.8795(0.0315)	0.9023(0.0426)	0.8828(0.0582)
Max Entropy	0.9152(0.0315)	0.9480(0.0448)	0.8937(0.0563)
SVM	0.9132(0.0357)	0.9262(0.0616)	0.9048(0.0433)

Table 2. Measurement on Data set 2. (Pos: Neg= 230:460)

	Accuracy	Recall	Precision
Naïve Bayes	0.8298(0.0371)	0.9611(0.0431)	0.6706(0.0487)
Decision Tree	0.9186(0.2031)	0.9089(0.0693)	0.8568(0.0469)
Max Entropy	0.9302(0.0245)	0.9263(0.0582)	0.8762(0.0677)
SVM	0.9157(0.0418)	0.8915(0.0900)	0.8662(0.0856)

And then using the classifiers trained by Data set 1 and 2, test on the test set. The results are shown in Table 3, and 4.

Table 3. Test set evaluation of Classifiers (Data set 1)

	Accuracy	Recall	Precision
Naïve Bayes	0.78	0.97	0.277
Decision Tree	0.921	0.974	0.522
Max Entropy	0.919	0.987	0.516
SVM	0.915	0.979	0.502

Table 4. Test set evaluation of Classifiers (Data set 2)

	Accuracy	Recall	Precision
Naïve Bayes	0.798	0.966	0.294
Decision Tree	0.941	0.97	0.595
Max Entropy	0.947	0.983	0.621
SVM	0.938	0.987	0.581

According to Table 1 to 4, we find that although the normal classifiers have a very good performance on the under sampling training set, when apply the classifiers to the test set the performance (see the column of precision) is decreased significantly. The reason may because the under sampling training set didn't represent all the features of the data to different the class "accept" and "reject".

Next we will test the performance about Active learning SVM. Because of the high dimensionality of the data, we use linear kernel rather than other non-linear kernels. Figure 6 is the learning curve of the Active Learning SVM, querying 20 instances every cycle, with initial sample size 10. The purple line is test set accuracy, blue line is training set accuracy, cyan line is the recall of the test set, and green line is the precision of the test set. The curve shows that the 8th output of the Active Learning SVM classifier is stable enough for the test set and training set. At that time only 170 instances are needed.

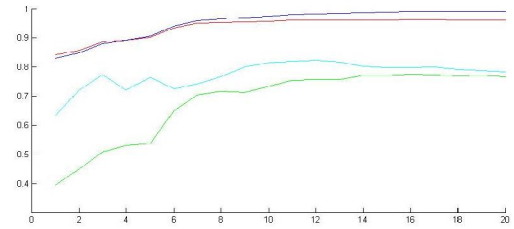


Figure 6. Learning curve (query 20 per cycle)

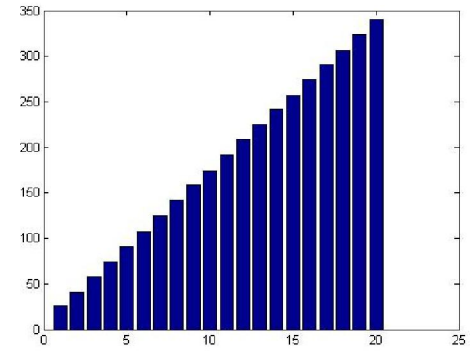


Figure 7. The number of support vectors changes

Figure 7 shows that the number of support vectors increases around 15 per cycle, which means among 20 queried instances, at least 15 instances are selected to be support vectors.

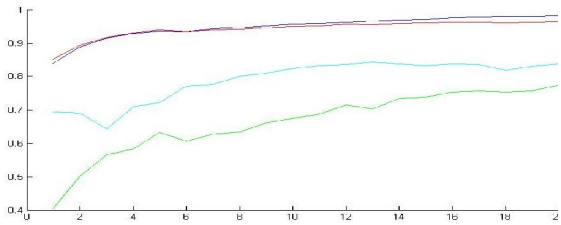


Figure 8. Learning curve query 10 per cycle

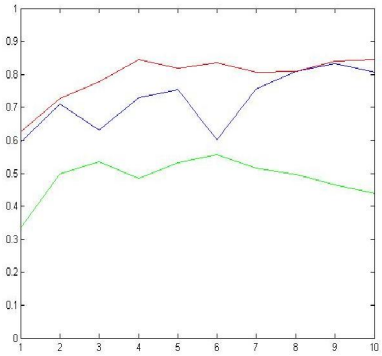
Figure 8 shows the learning curve for the Active Learning SVM, when querying 10 instances per cycle. The 20th output is good enough for the classification, which needs 210 instances and 20 times training for SVM. Comparing with Figure 5, there is not a significant improvement if we use the SVM with querying 10 instances per cycle. So we choose query 20 per cycle for this project.

Then we compare the measurements of the five classifiers on the test set. The result training set are Data set 2 (Pos:Neg = 230:460), and the test set are half of the whole data set (1368 instances). When the data is imbalance, a high accuracy doesn't mean a good performance, and we should focus on recall and precision. Because we use almost all of the positive instances as training data for

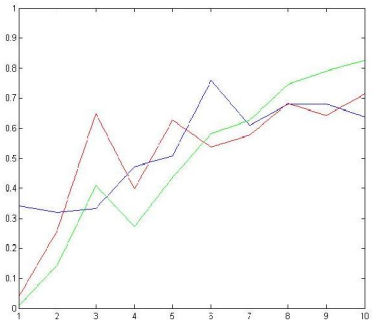
normal classifiers, so the recall is high. But when looking at precision, we find a around 15% improve on the Active Learning SVM classifier, which means that 73.6% of the data labeled as "accept" are ground truth label. That's a significant improve compare to other classifiers.

Table 5. The result based on test set

	Accuracy	Recall	Precision
Naïve Bayes	0.798	0.966	0.294
Decision Tree	0.941	0.97	0.595
Max Entropy	0.947	0.983	0.621
SVM	0.938	0.987	0.581
SVM Active Learning(8^{th} output)	0.956	0.776	0.736



(a) Precision learning curve



(b) recall learning curve

Figure 9. Learning curve for normal classifiers (Red: Max Entropy Blue: SVM Green: Naïve Bayes)

Figure 9 shows the precision and recall learning curve when changing the size of training data from 10% to 100%. We know for Active Learning SVM, to achieve a good and stable performance, it only needs 170 instances (8^{th} output), which is less than 10% of the whole data, while for the normal classifiers, shown in the Figure 9, a 70% percent of training data is needed to guarantee a good classifier. So Active learning does working better when the labeled data is expensive or hard to get than the normal classifiers.

Ⅴ. Conclusion

In this article, we introduce Active Learning SVM to the blog recommendation problem, and compare the performance to several normal methods (Naïve bayes, Decision Tree, Max Entropy, SVM). The result shows that the Active learning method is doing well on query samples when the data is imbalanced. The Active Learning SVM will decide which unlabeled instances should be queried and labeled, rather than random select from the pool. That gives a more active way of sampling. We also show that the Active Learning SVM just need about 10% of the training set to get a stable and good enough classifier, while others need more than 70%.

For this project, we still can try to choose other query methods, and then make the active learning more efficient. And also we may find a way to update SVM based on the previous SVM. What's more we may see the data as a time evolving data stream, which means the topics would be changed along with the time eclipse. The articles that would be accepted today do not mean it will be accepted later. If true, our classifier should be able to deal with it. Last we should find a way to estimate the error of the classifier, when the training data size is very small.

Reference

[1] King-Shy Goh, Edward Y. Chang, and

Wei-Cheng Lai. 2004. Multimodal concept -dependent active learning for image retrieval. *In Proceedings of the 12th annual ACM international conference on Multimedia(MULTIMEDIA '04).* ACM, New York, NY, USA, 564-571.

[2] Simon Tong, Daphne Koller. Support Vector Machine Active Learning with Application to Text Classification. *Journal of Machine Learning Research*, (2001) 45-66.

[3] Burr Settles. Active Learning Literature Survey. *Computer Sciences Technical Report* 1648, University of Wisconsin–Madison. 2009.