Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 30 (2014) 50 – 59

Procedia Computer Science

1st International Conference on Data Science, ICDS 2014

When a Classifier Meets More Data

Zhicheng Liao[a], Yangyong Zhu[a]

[a]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University Zhangjiang Campus, No. 825 Zhangheng Rd, Shanghai 201203, P. R. China

Abstract

The studies of generalization error give possible approaches to estimate the performance of a classification. But they are still expensive and difficult to use on large-scale data. In this paper, we discover that the accuracy of a classification is regional convergence with respect to the size of training data set, and give a Bounded Accuracy Conjecture. We also find that to train a classification with a little noisy training data set will not impact the accuracy. Finally, we give an easy but effectively experimental approach to build a good enough train data set for a given large-scale problem.

Keywords: classification; accuracy; regional convergence; large-scale data set; experimental approach

1. Introduction

Classification is an important research branch of data mining. "Accuracy" (or "precision") is a widely-used measurement for evaluating classifiers. In traditional point of view, a good training data set leads a high accuracy. But Banko and Brill [1] point out that comparing to the best system trained on a standard training data set, a worse classification will get benefit from more additional training data, and can get a higher accuracy than the best system. It seems that adding more training data to the training data set is a simple but effective way to increase the accuracy.

In the big data era, data is easy to obtain or generate. But it doesn't mean that more training data are easy to get. It is expensive to correctly identify category membership for items. In most time we can only annotate them manually. Furthermore, train a classifier by too many training data always requires more processing time and extreme large RAMs. On the other hand, if a classifier was trained by a large-scale but low-quality training data set, the classifier may useless in predicting stage, as one of our experiments shows.

Will a classifier work perfectly if we trained it with a large enough training data set? Unfortunately, we cannot give a guarantee for that even if we have a prefect (both in size and quality) training data set. As our study shows, training with more data does not certainly improve the accuracy. In many cases, training a classifier by more training data has little benefits but more costs. Studies of generalization error give a positive answer that the limit of the size of training data set exists. So we interest in the trend of the accuracy when the size of training data set is breaking through its limit. We focus on accuracy rather than generalization error because we

doi:10.1016/j.procs.2014.05.380

don't care about classification comparison that generalization error always inspects. We also omit impacts of the executing duration and the requirement of storages (i.e. the RAM and the disk space). Because they are in its blood, and do not depend on data.

We have three contributions in this paper: 1) we give out the Bounded Accuracy Conjecture that the accuracy of a classification is regional convergence with respect to the size of training data set; 2) we find that it will not impact the accuracy of a classification if it is trained with a training data set whose distribution has only a little different from actual data set; and 3) we give an easy experimental approach to train and predict for a given large-scale problem (i.e. data set is very large or infinite).

The rest of this paper is organized as follows. In section 2 we review some related research work. In section 3 we introduce our experiments in detail, and analyze those experimental results. In Section 4 we try to analyze the problem we interest, and bring out our conjecture. In section 5 we give an experimental approach for a given large-scale problem. The section 6 is the conclusion and the future work.

2. Related Work

In most studies of classification, we can find a rising accuracy curve with respect to the size of training data set. Banko and Brill [1] have studied five different Nature Language Processing learners, trained them up to one billion words of training data. They note that the worst learner trained on 20 million words outperforms the best learner trained on 1 million words. Fuxman et al. [2] also show out that the classifier can have accuracy gains by expanding the size of the manually labeled training set from 1K to 20K, and can have large accuracy gains by using a 1.5M automatically extracted training set. The cost of using automatically extracted training set is much low. It seems a happy result that the more training data if we have, the more accuracy we can always get.

Some researchers have doubts to it. Dumais, Banko, Brill, Lin and Ng [3] give some experiments on a web question answering system. The experimental results show that by using redundancy of large corpora to simplify the query rewrites, the question answering accuracy can be greatly improved by analyzing more and more matching passages. But the accuracy will falling off after more snippets are included for n-gram analysis without the simplification step. Reported by Machlis [4], Nate Silver, a famous American statistician, also believe that big data may seem to promise big insights to users, but more isn't always better. More data there is, the more people can cherry pick data points that confirm what they want it to show. In addition, most of classification algorithms are in-memory processing. It requires faster CPU and more RAMs, and maybe requires exponentially more time to train a classifier with more data.

Halevy, Norvig, and Pereira [5] use a big data strategy to learn from text corpus at web scale, and get a great success in natural language processing and related fields. They represent all the data with a nonparametric model, then use unsupervised learning on unlabeled data which is so much more plentiful than labeled data. In their approach, having more data is definitively much better for results. The benchmark results of the well-known PASCAL VOC object challenge [6] have shown a clear trend in increased performance over the years as methods have gotten better and training data sets have become larger [7]. But after they analyzed their experimental results, they suggest an alternative view, i.e. none of the models they tested will benefit greatly from more data. Instead, the largest gains were in enforcing richer representational structure and constraints within the model.

Studies on generalization error have similar opinions in theory and practice, as show in [8]. Niyogi and Girosi [9] point out that for Radial Basis Functions, after a critical number of nodes, overfitting occurs and the generalization error increases. They believe it apply to any approximation technique. In order to estimate generalization error, some resampling test set approach are given. Anthony and Holden [10] discussed in cross-validation approach. Bylander [11] gives an out-of-bag estimator on two-class data set. Moreover, bootstrap techniques [12] are frequently used in generalization error estimating. With study of generalization error

estimating, some researchers pay attention to constructing a confidence (training) data set for the generalization error.

3. Experiments

3.1. Experiment environments

We run all of experiments on a single PC server with hardware configurations as: E5640 2.66GHz*2 CPU, 32 GB RAM, and 3*146GB HDD. The operating system is 64-bit version of RedHat Enterprise Linux server 5.5. For running classifications which implemented in Java codes, we use JDK build 1.6.0_24-b07 64-bit version as Java runtime environment.

3.2. Classifiers

To ensure the correctness of algorithm implementations, We choose two famous data mining toolkits and pick up some classifications. One is libSVM [13] version 3.16, a library for Support Vector Machines (SVMs). It has built-in training tool (svm-train) and predicting tool (svm-predict). We choose widely-used radial basis function $exp(–gamma*|u-v|^2)$ as the kernel function. We denoted this SVM classification by SVM-C. The other toolkit is WEKA [14] version 3.5, a data mining software written in Java. We choose Naïve-Bayes, BayesNet, J48 (a clone of C4.5), MultilayerPerceptron, VotedPerceptron, and Logistic classifiers for our experiments, and most parameters of models are default values. Error penalty parameter *cost* and the kernel parameter *gamma* for SVM-C are determined for each different training data set by the proposed method which presented by Y. J. Ding, Z. G. Yan, and Z. Gao [15].

3.3. Data sets

We select two coupled binary-class data sets for experiments. Without loss of generality, all attributes except the class label are numeric. A format conversion preprocessing will be performed if needed.

One coupled data set is *cod-rna* and *cod-rna.t* [16]. It includes two classes, labeled as -1 (positive class) and 1 (negative class). The train data set *cod-rna* contains total 59,535 data items, which include 39,690 positive class items and 19,845 negative class items. The test data set *cod-rna.t* contains total 271,617 data items, which include 181,078 positive class items and 90,539 negative class. This data set has 8 features.

The other coupled data set is *kddb* and *kddb.t* [17]. The original data sets are from KDD CUP 2010. It includes two classes, labeled as 1 (positive class) and 0 (negative class). The train data set *kddb* contains total 19,264,097 data items, which include 16,579,660 positive class items and 2,684,437 negative class, i.e. 86.065% positive class items and 13.935% negative class. The test data set *kddb.t* contains total 748,401 data items, which include 664,374 positive class items and 84,027 negative class. This data set has 29,890,095 features.

3.4. Sequences of training data sets

Based on coupled *cod-rna* and *cod-rna.t*, we build 5 sequences, named *cod-rna.list*, *cod-rna.n0.05*, *cod-rna.n0.2*, *cod-rna.random* and *cod-rna.t.special*. In *cod-rna.list*, *cod-rna.n0.05*, *cod-rna.n0.2* or *cod-rna.random*, items of subsets are 100, 500, 1000,..., 59,500. *cod-rna.list* is an average sampling sequence of subsets from *cod-rna*. *cod-rna.n0.05* is based on the average sampling, with a 5% noise items. *cod-rna.n0.2* is similar to *cod-rna.n0.05* except the noise is 20%. *cod-rna.random* is a random sampling sequence of subsets from *cod-rna*. The *cod-rna.t.special* is generated from *cod-rna.t* which start from 500 items, extend with a step of 500 items, stop at 271,500 items, average sampling.

Generated from *kddb*, we build another 5 sequences, named *kddb.list*, *kddb.n0.05*, *kddb.n0.2*, *kddb.random*, *kddb.small*. In *kddb.list*, items of subsets start from 100, extend with a step of 500 items, stop at 100,000. We also adjust the size of the second subset to 500 so sizes of subsets (except the first) are multiples of 500. In *kddb.n5*, *kddb.n20* and *kddb.random*, items of subsets are 100, 1000, 2000,…, 100,000. In *kddb.small*, items of subsets are 1, 5, 10, 15,..., 100. *kddb.list* is an average sampling sequence of subsets. *kddb.n0.05* is based on the average sampling, with 5% noise items. *kddb.n0.2* has 20% noise items. *kddb.random* is an random sampling sequence. *kddb.small* is average sampling too.

3.5. Bounded accuracy experiments

We train SVM-C with *cost=3.45* and *gamma=0.001* by *cod-rna.list*, *cod-rna.n0.05* and *cod-rna.random*. We also run SVM-C with *cost=4.5* and *gamma=0.0322* on *kddb.list*, *kddb.n0.05*, *kddb.n0.2*, *kddb.random* and *kddb.small*. Fig. 1 shows accuracies with respect to the size of training data set when SVM-C trains by *cod-rna.list* and *cod-rna.n0.05*. Training by the average sampling sequence *cod-rna.list* has a threshold (denoted as *k*), when the size of training data set is greater than *k*, the accuracy fluctuates in a narrow interval. From the result we can find that the accuracy is bounded, *k=36500*, and $89.4\%<Accu(SVM\text{-}C_{Ti}, P)<90.3\%$. Training by a 5% noisy training data set appears a similar result.

Fig. 2 shows accuracy curves of SVM-C trains by *kddb.list* and *kddb.n0.05*. The accuracy of SVM-C training by *kddb.list* is bounded too, *k=100*, $88.3\%<Accu(SVM\text{-}C_{Ti}, P)<88.8\%$. But unlike the previous experiment, we note that the accuracies of two are almost same everywhere. Because when SVM-C classifier is under-fitted, it will simply predict all data to the majority class, i.e. the positive class in this case. So the accuracy will not less than the ratio of positive class 664374/748401=88.7725%. The zoomed-in part of curve as Fig. 3 shows. When size is greater than 100, the maximum accuracy is closed to the under-fitted accuracy. It explains why the classifier seemingly has a low threshold.

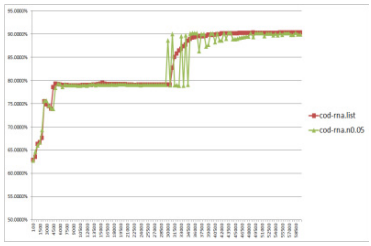

Fig. 1. Accuracy curves of SVM-C trains by *cod-rna.list* and *cod-rna.n0.05*.

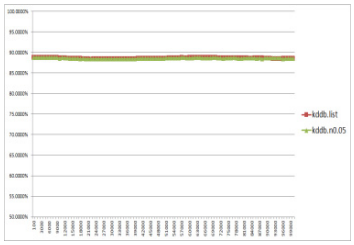

Fig. 2. Accuracy curves of SVM-C trains by *kddb.list* and *kddb.n0.05*. The curve of *kddb.n0.05* almost overlaps on *kddb.list's*.

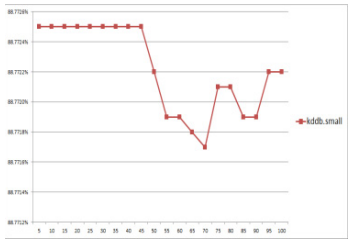

Fig. 3. The accuracy of under-fitted SVM-C trains by *kddb.small*.

We run those WEKA provided classifiers on *cod-rna* because perform them on *kddb* might be interfered by under-fit. Another reason is the *kddb* contains too many features that those classifiers may not handle without a feature selection filter.

Fig. 4 shows the accuracy of Naïve-Bayes and the accuracy of Logistic trained by cod-rna.list. It is obvious that the accuracy of Naïve-Bayes is bounded, *k=21500*, $77.0\%<Accu(Naïve\text{-}Bayes_{Ti}, P)<77.5\%$. The accuracy of Logistic is bounded too, *k=34500*, $92.5\%<Accu(Logistic_{Ti}, P)<95.2\%$. In further, we run the Naïve-Bayes classifier by using supervised discretization to process numeric attributes ("-D" command-line parameter), denoted by Naïve-Bayes-D. Fig. 5 shows that accuracies of Naïve-Bayes-D and Naïve-Bayes have the same boundary, *k=21500*.

3.6. Noisy distribution experiments

With our observation in experiments of using SVM-C classifier, a little noise to the distribution of training data sets does not significantly impact to the accuracy. Here we do more experiments to check it.

The accuracy curves of Trained by random sampling show in Fig. 6 and Fig. 7. The accuracy has no connection to the size of training data set, and curves are quite different from average sampling sequence.

We perform a 20% noise to the average sampling on *kddb*, and consider the accuracy of training by *kddb.list* as the baseline. As Fig. 8 shows, the accuracy curve of *kddb.n0.2* is more fluctuating than *kddb.n0.05*. On other words, the more noisy the training data set has, the less correction the classifier predicts. However, the accuracy of *kddb.n0.2* is the worst in these three results, it still varies between 87.7% and 88.8%. It is much better than the random sampling case. This fact also exists on the Logistic classifier. Fig. 9 shows the accuracy of *cod-rna.n0.2* versus the accuracies of the sequence *cod-rna.n0.05* and *cod-rna.list*. Unsurprisingly, the accuracy is positive related to the fitness of the distribution. The worst accuracy curve of the sequence *cod-rna.n0.2* is also much better than the random sampling sequence *cod-rna.random*.

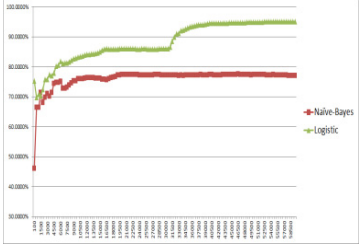

Fig. 4. Accuracy curves of Naïve-Bayes and Logistic train by *cod-rna.list*.

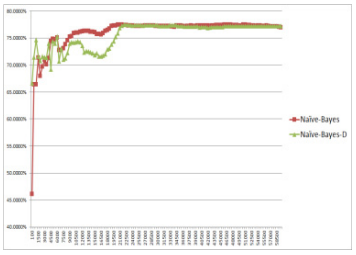

Fig. 5. Comparing the accuracy of Naïve-Bayes-D with Naïve-Bayes.

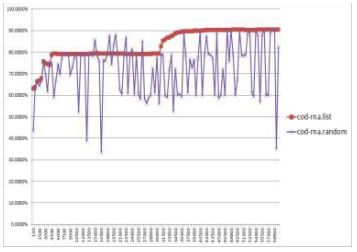

Fig. 6. The accuracy of SVM-C trains by *cod-rna.random* (vs. *cod-rna.list*).

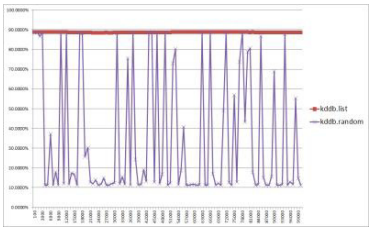

Fig. 7. The accuracy of SVM-C trains by *kddb.random* (vs. *kddb.list*).

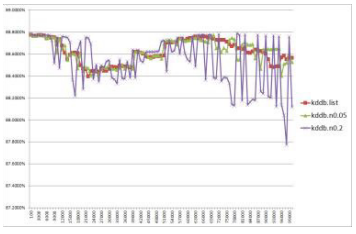

Fig. 8. The accuracy of SVM-C trains by *kddb.n0.2 (vs. kddb.list* and *kddb.n0.05)*.

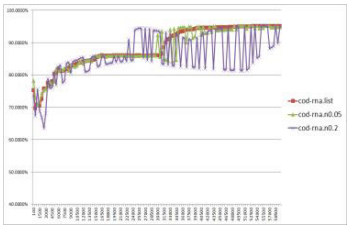

Fig. 9. The accuracy of Logistic trains by *cod-rna.n0.2* (vs. *cod-rna.list* and *cod-rna.n0.05*).

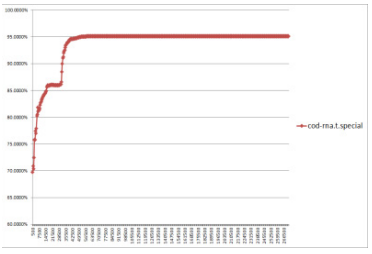

Fig. 10. The accuracy of Logistic trains by *cod-rna.t.special*.

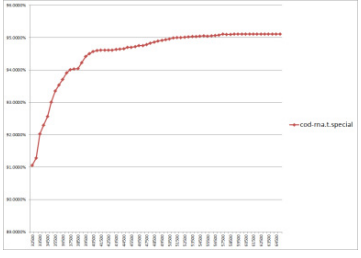

Fig. 11. A zoomed-in part of the accuracy of Logistic trains by *cod-rna.t.special* (size of T_i from 32500 to 65000).

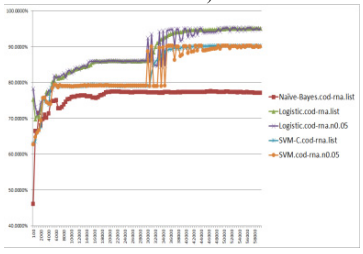

Fig. 12. Convergence intervals of SVM-C, Naïve-Bayes and Logistic.

3.7. Accuracy factor experiments

cod-rna.t.special is average sampling from the testing data set *cod-rna.t*. We expect that the accuracy will close to 100%. Fig. 10 shows the accuracy curve of Logistic which trains by *cod-rna.t.special*, and Fig. 11 is a zoomed-in part of the accuracy. Unfortunately, the accuracy is no increase after exceed 95.1119% at size of training data set over 60000. It's more clearly in Fig. that the curve is horizontal at the tail. Comparing with the accuracy of sequence *cod-rna.list* one by one, the accuracy is exactly equal. The *cod-rna.list* has the same distribution of *cod-rna.t.special*, but data items are not same. If we train by *cod-rna.t* and then test on it, the accuracy $Accu(Logistic_P, P)$ is incredible 95.5135% which is never in sight. Therefore, we believe it in enough reasons that the accuracy of a classifier depend on the distribution of training data set, not data items.

Another interest phenomenon we observed is every classifier has its own convergence interval. The classification algorithm determines the boundary of the convergence interval. A little noise of sampling training data will not change the convergence interval. In Fig. 12, the convergence interval of Logistic trained by noisy average sampling tends to the convergence interval of Logistic trained by restrict average sampling. It happens in SVM-C too. We surely believe it will happen to most of classifiers. Referring to Fig. 5, two accuracy curves which are based on Naïve-Bayes are overlapped with respect to the size of training data set. We believe that a minor improve of a classifier will not really impact the convergence interval. In addition, by careful studying values of accuracies in SVM-C experimental results, a classifier has different convergence intervals on different testing data sets. We guess that the length of convergence interval should be less than 0.02 (2%) in most cases.

3.8. Experiments on other classifiers

To further validate that the accuracy is bounded, and the threshold of the training data set size exists, we select other classifiers of different families, train them by the average sampling training data set sequence *cod-rna.list* and the little noisy average training data set sampling sequence *cod-rna.n0.05,* and validate them by *cod-rna.t*. The mean reason we give up using *kddb* and *kddb.t* is *kddb* and *kddb.t* contains too much attributes and most selected classifiers will be over-fitted without performing a feature selection. In addition, for some classifiers, sizes of *kddb* and *kddb.t* are too large to complete in a tolerable time.

Accuracy curves of BayesNet trained by *cod-rna.list* and *cod-rna.n0.05* show in Fig 13. In Fig. 13, both accuracy curves vary largely at the beginning, but with the increasing of the training data size, they tend to stable and approaches to the optimum. The accuracy of BayesNet trained by *cod-rna.list* is bounded, $k=21500$, and $76.9\%<Accu(BayesNet.cod\text{-}rna.list_{Ti}, P)<77.5\%$. Furthermore, the accuracy freezes at 77.2109% when the size is greater than 50000. The accuracy of BayesNet trained by *cod-rna.n0.05* is similar to which trained by *cod-rna.list*, but $k=49000$, and $77.1\%<Accu(BayesNet.cod\text{-}rna.n0.05_{Ti}, P)<77.6\%$.

J48 is a clone of C4.5, which is belong to decision tree family. Accuracy curves of J48 show in Fig. 14. In Fig. 14, accuracies increase with the increasing of the training data size. Obviously, the accuracy of J48 trained by *cod-rna.list* is bounded, $k=50500$, and $94.8\%<Accu(J48.cod\text{-}rna.list_{Ti}, P)<95.2\%$. The accuracy of J48 trained by *cod-rna.n0.05* is bounded too, $k=54000$, and $94.1\%<Accu(J48.cod\text{-}rna.n0.05_{Ti}, P)<95.3\%$.

MultilayerPerceptron (MLP) uses back propagation to classify instances. The accuracy curves of MLP trained by *cod-rna.list* and *cod-rna.n0.05* show in Fig. 15. In Fig. 15, both accuracies tend to regional convergence after a certain size of training data set, but run up and down from the beginning. The accuracy of MLP trained by *cod-rna.list* is bounded, $k=34000$, and $94.4\%<Accu(MLP.cod\text{-}rna.list_{Ti}, P)<96.3\%$. The accuracy of MLP trained by *cod-rna.n0.05* is also bounded in the final, $k=42500$, and $94.9\%<Accu(MLP.cod\text{-}rna.n0.05_{Ti}, P)<96.3\%$.

VotedPerceptron (VP) is another classifier which is based on the propagation algorithm. The accuracy curves of VP trained by *cod-rna.list* and *cod-rna.n0.05* show in Fig. 16. Accuracies increase quickly, and keep stable. The accuracy of VP trained by *cod-rna.list* is bounded, $k=9000$, and $88.1\%<Accu(MLP.cod\text{-}rna.list_{Ti}, P)<89.1\%$.

The accuracy of VP trained by *cod-rna.n0.05* is also bounded in the final, $k=42500$, and $88.7\%<Accu(MLP.cod\text{-}rna.n0.05_{Ti}, P)<89.1\%$.

According to these experimental results in this subsection, we find all accuracies are bounded. In many results of experiments, we can find that training by a large training data set which the size is over the threshold k, a little noise of average sampling doesn't absolutely change the accuracy a lot. We put all accuracy curves which trained by *cod-rna.list* together and magnify the tail part, as Fig. 17 shows. For those Bayesian family classifiers, shapes of their accuracy curves are very similar. Especially, the Naïve-Bayes-D and BayesNet almost equal everywhere.

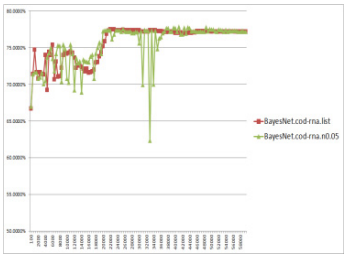

Fig. 13. Accuracy curves of BayesNet trained by *cod-rna.list* and *cod-rna.n0.05*.

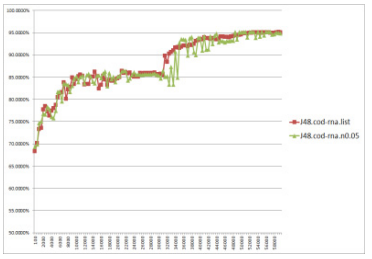

Fig. 14. Accuracy curves of J48 trained by *cod-rna.list* and *cod-rna.n0.05*.

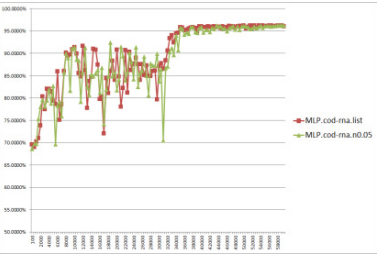

Fig. 15. Accuracy curves of MLP trained by *cod-rna.list* and *cod-rna.n0.05*.

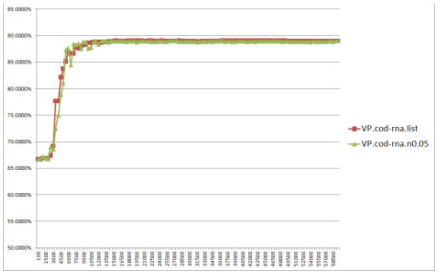

Fig. 16. Accuracy curves of VP trained by *cod-rna.list* and *cod-rna.n0.05*.

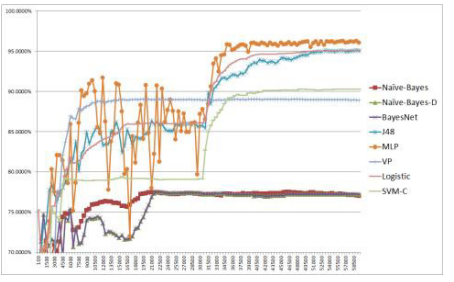

Fig. 17. Magnified accuracy curves of all classifiers of our experiments, trained by *cod-rna.list*.

4. Problem Analysis and Conjecture

We observed from experiments that the accuracy is bounded, and there exists a threshold k, the accuracy is regional convergence if the size of training data set is greater than k. Is this really acceptable? We will try to explore the fact. For easy to discuss, we always consider the training data set will not make the classification over-fitted (but under-fitted is allowed).

4.1. Preliminary analysis

In ideal case, if we could enumerate all items with categories which a classifier shall process, we can simply use a lookup method to correctly classify all data. Definitely, this classification will work perfectly and have 100% accuracy. In the mathematical form, for all n training items $T_n=\{a_1, a_2, ..., a_n\}$, we use T_n to train a lookup classifier L, then we use the trained classifier $L_{Tn}=Train(L, T_n)$ to predict (test) every item in testing data set $P=T$. All items in P is already recognized in the training stage (as in a dictionary), so the accuracy $Accu(L_{Tn},P)=Accu(L_{Tn},T_n)=100\%$. Unfortunately, we are difficult to make a such training data set. Because we cannot enumerate all data in most cases, this method is rarely useful in practice.

Then we give a weak ideal case. If the size of P is greater than 2, we can make a training data set $T_{n-1}=P-\{a_n\}$, i.e. T_{n-1} is a subset of P which only removes one item from P. After training L with T_{n-1}, there are only two possible results: L_{n-1} can correctly predict all a_i, the accuracy is still 100%; or it can correctly predict a_1 to a_{n-1} but wrong to predict a_n, the accuracy is $(n-1)/n<100\%$. So the accuracy satisfies $Accu(L_{T\,n-1}, P)\leq 100\%$.

Let's decrease the size of training data set T_{n-2} by removing one item a_{n-1} from T_{n-1}, i.e. $T_{n-2}= T_{n-1}-\{a_{n-1}\}$. To iterate these steps until to $T_1=\{a_1\}$, we can get a sequence of training data sets $T_1 \subset T_2 \subset \ldots \subset T_n$, and

$$Accu\ (L_{T1}, P)\leq Accu\ (L_{T2}, P)\leq \ldots \leq Accu(L_{Tn}, P)\leq 100\% \tag{1}$$

It means in the ideal case, the accuracy is monotonic increasing with respect to the size of training data set, and will reach to the theoretical upper bound (100%), as (1) shown. It is exciting but almost unreal.

For an representative-based or probability-based approximation classification C,

$$Accu(C_{Ti}, P)\leq Accu(L_{Ti}, P) \tag{2}$$

We use the accuracy decrease coefficient r_i, $0\leq r_i\leq 1$ to rewrite (2) as $Accu(C_{Ti}, P)=r_i*Accu(L_{Ti}, P)$. If r_i is a constant or monotonic increasing with respect to $|T_i|$, we can infer (1) to:

$$Accu(C_{T1}, P)\leq Accu(C_{T2}, P)\leq \ldots \leq Accu(C_{Tn}, P)\leq 100\% \tag{3}$$

But if r_i is a random parameter or non-monotonic, (3) is false.

To note that (2) is always true whatever the characteristic of r_i is, we can regard $Accu(L_{Tn}, P)$ as the upper bound of the accuracy of a classification C. Obviously, 100% is a trivial upper bound for any classification.

On the other hand, a well-trained classification algorithm can always correctly predict data items which is matched the properties of the training data set. Here the properties refer to the distribution of a data set, functional dependency of attributes, etc. Therefore, the accuracy of a trained classification C_{Ti} should not less than an associated value, i.e.

$$b_i\leq Accu(C_{Ti}, P) \tag{4}$$

In most time, b_i should not be less than $1/m$ which m is the total number of classes.

We regard the b_i as the lower bound of the accuracy of a classification C. It's also obvious that 0% is a trivial lower bound for any classification. With (3) and (4), the accuracy of any classification is bounded.

4.2. Bounded accuracy conjecture

We notice that for a given classification, the accuracy (with respect to the size of training data set) fluctuates after a specific size. In other words, we declare a conjecture about the accuracy of a classification (on a certain data set) as follows.

Bounded Accuracy Conjecture: for any classification C and a given sequence of training data sets $T_1 \subset T_2 \subset \ldots \subset T_n$, there exists an integer threshold k, that for any trained classifier C_{Ti} which the cardinality of training data set $|T_i|>k$, there exists

$$b_k\leq Accu(C_{Ti}, P)\leq a_k,\ 0\leq b_k\leq a_k\leq 1 \tag{5}$$

The b_k is the infimum of the accuracy of C, and a_k is the supremum (under the given sequence of training data sets). For all possible sequences of training data sets, the greatest $b_k=b$ is the worst accuracy and the least $a_k=a$ is the best accuracy of classification C.

With the Bounded Accuracy Conjecture, for any classification C which uses approximation method, which uses approximation method, and trains by $T_i \subset T_{i+1} \subset \ldots \subset T_n$, $k<|T_i|<|T_{i+1}|\ldots<|T_n|$, the accuracy of C_{Tj} ($j=i, i+1, \ldots, n$) is in a non-zero-length interval $[b_k, a_k]$, and the condition in (5) is restricted to $0<b_k<a_k\leq 1$.

5. An Acceptable Experimental Approach for Solving a Given Large-scale Problem

Let's think about predicting in a runtime system. If we can discover the distribution of the given data set, it is easy to build a good enough training data set by average sampling and well labeling. Although we can use statistical methods to estimate the distribution, but in most cases, we cannot get an accurate answer. Moreover, if data is real-time generated, we are difficult to know the class label of the next data. Other classifier performance estimating methods, for example k-fold cross-validation, are not applicable too. However, we still have an experimental approach to build a good training data set without knowing the distribution of the given data set.

According to our Noisy Distribution Experiments, training by a little noisy (5%) average sampling is more acceptable than a large noisy (20%). Regardless which classifier is, the accuracy curve of little noisy average sampling almost overlap on the baseline i.e. the accuracy curve of the average sampling. So we can give an approach to build a good training data set for a given large-scale problem.

Here we assume that we can get enough data from the runtime system. First, we freely collect some data to build an initial training data set. We recommend the initial training data set contains hundreds of data, although it can be in any size. We manually annotate the initial training data set with care. Second, we train a classifier by the training data set and validate in the runtime system. We can use a simply and fast classifier, for example, Naïve-Bayes, to save training time. It doesn't affect us to build a good training data set. We calculate the accuracy continuously until to a stable value. Third, we add some data of one class to the training data set. For example, all added data belong to positive class. Then we repeat the second step with the new training data set and get the new accuracy. If the new accuracy is greater than the old one, we keep on adding data of the same class; if the accuracy is less than the old, we add data of another class (for example, the negative class). Repeat the third step in several times, and observe varies of accuracies. If the accuracy jumps in a large range, the third step will be done more times. We should repeat these steps until the accuracy fluctuates in a narrow acceptable interval. If the accuracy convergence interval appears, the training data set is built. In addition, we can verify it by training another classifier and testing in runtime system as second step. The only note is to keep a low redundancy on every training data set.

Nowadays, it is popular to using the cloud computing architecture to deal with large-scale data. Traditional runtime feedback approaches are helpful to the performance of a classification, but it is not suitable in the cloud computing situation. Because it is impossible (and no reasonable) to synchronize feedbacks to all instances of the classification which run on distributed computing nodes. This is a disadvantage to running a classification on the cloud computing architecture. In our experimental approach, if real data is generated infinitely and dispatched to computing nodes without discrimination, we can periodically calculate the distribution of predicted data, and rebuild a new incremental training data set to re-train the classification dynamically when the distribution changes a lot from the last state. The distribution of the updated training data set fits to current real data, so we can make classification running on a good performance at any time.

6. Conclusion and Future Work

In this paper, we try to explore some key factors which impact a classifier to have a high accuracy in the general binary classification problem. By our studies, the distribution of the training data set is very important. It

does not significantly impact the accuracy if the distribution of training data set is a little different from the target data set. For a given problem, there exists a threshold of the size of training data set, when training by any oversized training data set, the accuracy with respect to the size of training data set is regional convergence. Based on these conclusion, we bring out an acceptable experimental approach to a given large-scale problem. It's easy to implement and work effectively on a cloud computing architecture.

Our study has some limitations. For example, we only study on binary classification problems, and have not verified on all kind of classification families. In the future, we will verify our conclusions on the general multi-class classification problem.

Acknowledgements

This work is supported in part by the National Key Technologies R&D Program Grants No. 2012BAH13F02 and the key scientific and technological projects of Science and Technology Commission of Shanghai Municipality Grants No. 12511502403, Grants No. 12511509602.

References

[1] Banko M, Brill E Mitigating the paucity-of-data problem: exploring the effect of training corpus size on classifier performance for natural language processing. Proc. of the first intl. conf. on human language technology research; 2001, p. 1-5.
[2] Fuxman A, Kannan A, Goldberg AB, Agrawal R, Tsaparas P, Shafer J. Improving classification accuracy using automatically extracted training data. Proc. of the 15th ACM SIGKDD intl. conf. on knowledge discovery and data mining (KDD '09); 2009, p. 1145-1154.
[3] Dumais S, Banko M, Brill E, Lin J, Ng A. Web question answering: is more always better? Proc. of the 25th annual intl. ACM SIGIR conf. on research and development in information retrieval; 2002, p. 291-298.
[4] Machlis S. More data isn't always better, says Nate Silver. Computerworld magazine (online news); May 8, 2013. Retrieved from: http://www.computerworld.com/s/article/9239019/More_data_isn_t_always_better_says_Nate_Silver
[5] Halevy A, Norvig P, Pereira F. The unreasonable effectiveness of data. Intelligent systems, IEEE 2009;24(2):p. 8-12.
[6] Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A. The PASCAL visual object classes (VOC) challenge. Intl. journal of computer vision 2010;88(2):p. 303–338.
[7] Zhu X, Vondrick C, Ramanan D, Fowlkes C. Do we need more training data or better models for object detection? British machine vision conf. (BMVC) 2012.
[8] Anguita D, Ghelardoni L, Ghio A, Ridella S. Test error bounds for classifiers: a survey of old and new results. 2011 IEEE symp. on foundations of computational intelligence (FOCI); 2011, p.80-87.
[9] Niyogi P, Girosi F. On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. Neural computation, MIT press journals 1996;8(4):p. 819-842.
[10] Anthony M, Holden SB. Cross-Validation for binary classification by real–valued functions: theoretical analysis. Proc. of the 11th conf. on computational learning theory; 1998, p. 218-229.
[11] Bylander T. Estimating generalization error on two-class datasets using out-of-bag estimates. Machine learning 2002;48(1-3):p. 287-297.
[12] Jain AK, Dubes RC, Chen C. Bootstrap techniques for error estimation. IEEE trans. on pattern analysis and machine intelligence 1987;PAMI-9(5):p. 628-633.
[13] Chang CC, Lin CJ. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2011;2(3):p. 2:27:1-27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
[14] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA data mining software: an update. SIGKDD explorations 2009;11(1):p. 10-18.
[15] Ding YJ, Yan ZG, Gao Z. A Method for Selecting Parameter of SVM with RBF Kernel. Sciencepaper online 2010. Chinese version retrieved from: http://www.paper.edu.cn/releasepaper/content/201005-659
[16] Uzilov AV, Keegan JM, Mathews DH. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. BMC Bioinformatics 2006;7:173. Retrieved from LibSVM data download web page, cod-rna in LibSVM format, http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#cod-rna
[17] “Bridge to Algebra 2006-2007” (training data set) and “Bridge to Algebra 2008-2009” (test data set) provided by KDD CUP 2010 Educational data mining challenge, 2010. Retrieved from LibSVM data download web page, kdd2010 (bridge to algebra) in LibSVM format, http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#kdd2010%20(bridge%20to%20algebra)