Dynamic Model Selection without Prior Meta-learning

Sunita Sarawagi sunita@it.iitb.ernet.in
Shantanu Godbole shantanu@it.iitb.ernet.in
Indian Institute of Technology Bombay

Abstract

We present a new framework for *impromptu* selection of the best amongst specialized classifiers without any synchronized meta-learning phase. Such a need would arise when classifiers start being shared widely and freely like information on the internet. An ad hoc user will then need to select the best model without any knowledge of the training process. Typically a one-time user will have just the instance to be classified and no separate validation dataset.

We show how none of the existing model combination algorithms meet the demands of impromptu model selection. We propose a novel approach that relies only on model-neutral information collected dynamically at the time of classifying a instance. Analytical and experimental evaluation of the method show its superiority to existing voting schemes on a wide range of settings.

1. Introduction

Data mining is at the chasm of widespread deployment. A major hurdle in the popular use of mining is that the level of expertise, data and resources needed in building a robust model is only affordable by a few large organizations. One way to facilitate widespread use in spite of this limitation is to allow people to share each other's model. Such gregarious sharing is possible in non-competitive domains as facilitated by the internet. As an example consider a document classification scenario.

On the web today, there are several large categorized document repositories like Yahoo[1], Altavista[2] and Google[3]. Many of these follow similar document hierarchies. For example, the Directory Mozilla[4] (Open Directory Project) is followed by major search engines like Google, Hotbot, and Infospace. With their up to-date categorized data and hardware resources, these sites are in a position to build large document classifiers and make them available as internet services. A lay user without any expertise in model building like a newspaper agency, or a library can now automatically categorize their documents using these services. Thus the benefits of mining start accruing to common people without their spending the resources in maintaining an up to-date database and going through the highly specialized art of building a good classifier.

Several interesting research challenges will need to be addressed before this can be made a reality. These form part of our larger ongoing project (Sarawagi & Nagaralu, 2000). This paper addresses novel model combination problems that arise when multiple classification models start being offered for sharing. If the different models assign different categories to the same document, which prediction should be trusted more? Presumably different sites have their areas of strength and weakness and no single site will be constantly better than another. A naïve user just interested in a one-time use may not have a validation dataset with which to benchmark the available options. Even if he possessed such a dataset, he has no control over the component models. Sites might evolve their models at will or become unavailable. Therefore, any method relying on synchronized meta-learning may not be feasible. What we need is an *impromptu* model selection algorithm that dynamically for each instance selects the best combination of models. We investigate how a model can establish its prediction quality around a test instance submitted by an ad hoc user who had no role to play in its training process.

Existing model combination methods (Dietterich, 1997) are not applicable in this setting due to various reasons starting from centralized control of model training and assumptions about homogeneity of component models to reliance on a separate validation dataset. Section 2 discusses this issue further. In Section 3 we introduce our proposed method of impromptu model selection. We present analytical and

[1] http://www.yahoo.com
[2] http://www.altavista.com
[3] http://www.google.com
[4] http://dmoz.org

experimental evaluation of this method in Sections 4 and 5 respectively. We analyze the strengths and weakness of the proposed algorithm in Section 6 and conclude with a call for further investigation of this new paradigm of impromptu model selection in Section 7.

2. Existing model combination algorithms

We first summarize the requirements of a impromptu-selection method.

1. Heterogeneous-classifiers: Each site might independently decide to use a different classifier without revealing its method to the user.
2. No-global-synchronization: Classifiers might evolve by changing their training data or model at will. Therefore any method based, for example, on accumulated accuracy estimates or a synchronized meta-learning phase will not work.
3. No-validation-dataset: The model user is ad hoc and does not have a large validation dataset. Typically a user will have just the instance to be classified.
4. Specialized-classifiers: Each site acquires its training dataset independently and has its own areas of expertise. However, broadly all datasets are from the same domain and classifiers model the same input-output relationship.

The topic of combining several classifiers to produce a composite classification has received considerable attention in recent machine learning literature (Dietterich, 1997; Dietterich, 2000; Chan & Stolfo, 1995; Webb, 2000; Bauer & Kohavi, 1996; Prodromidis et al., 1997; Pennock et al., 2000). However, most of the existing model combination methods rely on a prior meta learning phase where a separate training dataset is used to learn the best model combination method. In many of these, for example, boosting, bagging, and stacking even the training data used for building the models is under centralized control. We know of no method that meets all of these requirements. There are however related methods that work when one or more of these requirements are waived.

The closest method in this genre is majority voting (MV). It satisfies all but the specialized-classifiers property. MV gives equal weightage to all classifiers and thus takes no cognizance of the relative expertise of the different classifiers.

A second genre of methods assume that each prediction is attached with a confidence or weight that indicates the coverage of the training data or prediction accuracy around the test instance. The winning class is then chosen by taking either the sum or the maximum of the confidence values per class. The method of finding the weights is different for different classification methods. When all sites are nearest neighbour classifiers, a natural choice is to base the weights on the average distance to the test instance as suggested by Tsymbal et al. (1998; 2000). Smaller distances imply higher likelihood that a site has seen data in the region of the input space and hence should be trusted more. Similarly for Bayesian classifiers, the posterior probabilities of classes can be used as a confidence estimate.

For neural networks a number of such weighting functions have been proposed. Weigend and Nix (1994) estimate confidence as the input dependent variance of the neural net. A new subnet, branched from the outermost hidden layer of the original network, is trained to output the variance around each prediction. Another possible measure of confidence is the activation level of a output node. Tresp and Taniguchi (1995) propose gating neural networks based on both the coverage of the input space close to the test instance and the variability of the model's prediction in that region. For estimating coverage they propose bringing together the training data of all sites and modeling a separate mixture distribution for the conditional probability of the test instance belonging to a site. This approach therefore also violates the no-global-synchronization property of impromptu-selection.

Koppel and Engelson (1996) propose a scheme of weighting *any* model by constructing at each site a separate decision tree that partitions its training space based on whether the model was correct or not in that region. A model's confidence is the purity of the leaf into which the test instance falls. This was one of the first methods that we had independently designed. It did not work well because leaves of different sites are incomparable — one site might get 100% purity just because it has few training instances in that region. In the same region another site might have significantly more data and therefore have lower than 100% accuracy. The second site's prediction should be trusted more even though it has lower accuracy at that leaf.

There are two positive aspects of the above weighting-based schemes. First, most (except (Tresp & Taniguchi, 1995)) do not require synchronization amongst classifiers. Second, the weights are determined dynamically for each instance based on the model's performance around the instance. In the literature these are referred to as dynamic model selection methods. This is in contrast to static combination methods that are based on precomputed performance measures like accuracy on whole training data as in Boosting (Schapire & Singer, 1999). All such dynamic methods therefore satisfy our specialized-classifiers property.

The main limitation of the above methods is that the weights are not comparable across different classifiers. For example, a nearest neighbour distance-based confidence value cannot be compared with a naive-Bayes classifier's posterior probability-based confidence. Thus, they violate the first condition of heterogeneous-classifiers.

A third genre of model selection methods are applicable under the heterogeneous-classifiers setting but require prior meta-learning. Merz (1998) proposes one such scheme. First, on a validation dataset predictions of all sites are collected. The matrix of instance by site prediction is subjected to correspondence analysis to find key prediction patterns. During classification of a test instance the processed matrix of validation data is consulted to find examples that had similar predictions from different sites. The model that performed best on these similar cases is chosen.

In the next section we propose a new model selection algorithm that builds on the dynamic methods discussed here but does so in a model neutral manner and without requiring a prior synchronization phase.

3. Proposed algorithm

Given an instance T, our prediction problem is to assign it to one of a flat set of k classes. The test instance is submitted in parallel to n sites. Each site outputs exactly one of these k classes. We propose a new Evidence-Boosting algorithm that at the time of classifying a test instance, holds additional interactions with the component classifiers to find the one site that is most relevant to T.

3.1 The Evidence-Boosting algorithm

In the Evidence-Boosting algorithm, first each site along with its predictions outputs *evidences* in the form of representative instances from its training data that are similar to T. This is akin to real-life situations where an expert backs his judgment quoting similar cases drawn from the past. A site by pulling out similar cases proves that it has been trained on relevant cases and should therefore be trusted more. We defer the issue of how a site would select such evidences to Section 3.1.2

The next issue is comparing the evidences of different sites in terms of their relevance to the test instance. We cannot rely on any measure output by each site because these may not be comparable across sites. The second option is to let the user define his own distance measure. However, designing a relevant distance measure is not easy. Finding a good distance measure on the attributes of the input instance can be as hard as the original classification problem.

We establish the proximity of a site's evidence to the test instance by consulting the other $n-1$ sites. Every time a site gives the same prediction to both the test and an evidence instance, it is implicitly voting for the relevance of that evidence, irrespective of the correctness of the prediction. Clearly, this relationship between proximity and belongingness to the same class is asymmetric. Whenever two instances are very close to each other they are likely to be assigned the same class label but within a class, instances might be far apart. By averaging over multiple sites we get a decent approximation. The user coordinates this conversation amongst the n sites as follows:

Evidence-Boosting:

1. User sends T to each of the n sites.
2. Each site i along with the predicted class c_i of T returns e evidences close to T from its training data for class c_i. Section 3.1.2 describes how each site picks its evidences.
3. User sends to each of the n sites the $(n-1)e$ evidences of its peers and collects their predictions.
4. For each evidence E, find the proximity with T by comparing the vector of n predictions of each. Section 3.1.1 describes the form of the proximity function.
5. Choose the site with the maximum average proximity over its e evidences.

3.1.1 Proximity function

We discuss how to establish the relevance of an evidence E to the test instance T. We treat the predictions of the n sites on T and E as two n dimensional vectors $\vec{P_T}$ and $\vec{P_E}$ respectively. We then deploy one of the many available vector similarity measures to compute the proximity $prox(\vec{P_T}, \vec{P_E})$ between these two vectors. One option is the fraction of sites where $\vec{P_T}$ and $\vec{P_E}$ agree. This ignores matches across sites. For instance if $\vec{P_T} = (1,2,3)$ and $\vec{P_{E_1}} = (2,3,1)$, $\vec{P_{E_2}} = (3,4,5)$ then both $prox(\vec{P_T}, \vec{P_{E_1}})$ and $prox(\vec{P_T}, \vec{P_{E_2}})$ will get a value of 0, although E_1 appears closer than E_2. We therefore also find the L_1 distance between the predicted class frequencies and convert the distance to proximity by scaling and subtracting from one. We then take an average of the two proximity values.

Thus each evidence gets attached with a proximity value calculated through democratic means. We sum these proximity values over all evidences of a site and choose the one with the largest sum. Ties are resolved by summing the site weights over all predicted classes and choosing the class with the largest total weight.

3.1.2 Choosing evidences

The next issue is picking evidences. An evidence is good if all sites give it the same prediction as the test instance irrespective of whether that prediction is correct or not. Thus, if the true class of T is A and 50% sites predict its class as A and another 50% predict class B, then a good evidence is one with the same 50-50 confusion amongst classes. In contrast, an evidence strongly about class A such that all classes give it the same prediction will be a bad evidence.

The selection of evidences is done individually by each site, therefore we can exploit classifier specific information in choosing the evidences. One option is for a site to design a new similarity function and choose the e evidences closest to the test instance based on this similarity function. However, we believe that an evidence selection mechanism that is closely tied to the site's classification model will fare better and also reduce the amount of extra work done during training. Our goal is to pick evidences that are close together in an implicit decision space used by a classifier. We show how this can be done for a few common classification methods:

Nearest Neighbor Model: Pick as evidence the closest e instances in the same class as the predicted class of the test instance.

Decision Tree: Return e representative instances from the same leaf of the decision tree as the test instance.

Bayesian classifiers: Find the posterior probability of the test instance belonging to each of the k classes. Measure proximity based on this k-dimensional vector and return the e closest ones.

Neural network: Map each instance to a vector corresponding to the activation level of the output units and the outermost hidden layer. Return evidences based on proximity in this space.

Similarly for other decision methods we can identify a few intermediate variables that capture proximity in the decision space. For example, if a site uses an ensemble classifier, it can compute proximity based on a larger vector formed by combining the variables of each component classifier.

A performance issue in the evidence selection phase is that we need to retain additional training data, which is an overhead for all classifiers except nearest neighbor. For decision trees we need to store e evidences for each leaf in the tree. For vector-based approaches as in Bayesian classifier and Neural Networks we reduce the amount of data to be maintained by clustering. After the classifier is trained, we map each training instance to its vector representation using the trained model and cluster these vectors. With each cluster we store only a fixed set of representatives. When a new test instance arrives, we choose from the stored representatives, those e instances that are closest to the test instance in the probability space and has the same class prediction.

Symbol	Explanation	Default
k	number of classes	5
n	number of sites	5
e	number of evidences	5
p	accuracy of each classifier	50%
N	number of test instances	500
s	skew amongst class priors	0.5
b	evidence bias	0.1

Table 1. Parameters with default values

The Evidence-Boosting algorithm thus achieves all our goals of impromptu-selection of models. It doesn't make any assumptions about the local classification method, doesn't require any prior meta-learning phase and doesn't collect any static information about classifiers. All extra information required for choosing the right class is collected dynamically and is totally within the local control of a site.

4. Understanding Evidence-Boosting analytically

We evaluate the performance of Evidence-Boosting in two parts. First we analyze the key characteristics of the algorithm by studying it under a simple setting and varying various parameters. Next in Section 5 we present results on real-datasets.

We considered a simple setting with k classes, n sites and e evidences. Each site's classifier was modeled as follows. For an instance with true class c_t, the classifier predicts the correct class c_t with probability p, and predicts one of the $k-1$ remaining classes with probability $(1-p)$, determined by a class prior. A parameter s controls the skewness amongst class priors. For each class, a Gaussian random number generator (with mean 0 and variance 1) is used to determine the prior of a class as $\frac{1}{k(1-s)^r}$. When $s = 0$ all classes have the same prior, when s is close to 1 there is maximum skewness across classes. Different sites choose their own priors this way. For each test instance T, each site returns e evidences. The proximity of the correct site's evidence to the test instance T is modeled as the probability that the evidence gets the same prediction as the test instance. We denote this as a parameter b.

We use the majority voting (MV) algorithm for comparison because it comes closest to meeting the requirements of impromptu-selection. For N test instances we measure the accuracy of Evidence-Boosting(EB) and the MV. Since both cannot give the right prediction when all n sites give wrong results, our performance metric is the number of cases where EB (or MV) is correct as a fraction of the cases where at least one classifier gives the right prediction.

Table 1 summarizes the parameters along with their default values.

Single classifier accuracy In Figure 1 we show accuracy as a function of p, the accuracy of a single classifier. EB is consistently better than majority for a wide range of values. The gap is larger for smaller values of p. An analytical explanation for this phenomenon is presented next. Consider evidences from two kinds of sites: E_c evidence of a site that made the correct prediction on T and E_w evidence of a site whose prediction on T was wrong. The true class of T and E_c is the same but E_w is different. Let us for the moment ignore the parameters s and b. The probability of a site predicting the correct class is p and one of the wrong classes is $\frac{1-p}{k-1}$. We define a term match probability p_c that denotes the probability of E_c getting the same class as T at a third site.

$$p_c = p^2 + (1-p)\frac{1-p}{k-1} \quad (1)$$

The first term p^2 is the probability that both T and E_c get the correct class and the second term is the probability of both getting the same wrong class. Similarly, the match probability p_w of E_w is:

$$p_w = p\frac{1-p}{k-1} + \frac{1-p}{k-1}p + (k-2)(\frac{1-p}{k-1})^2. \quad (2)$$

Here the first term is the probability that T gets the correct class and E_w gets the same prediction as T, second term is where E_w gets the correct class and T the same class as E_w and the third term is where both get one of the remaining $k-2$ classes.

In Figure 2 we plot these two match probabilities as a function of p. For both large and small values of p, E_c's prediction has a higher chance than E_w's of matching with the prediction on T. For p around $\frac{1}{k}$, both evidences have similar proximity values, making it less likely for the correct site to win. Figure 2 also shows the accuracy of EB with evidence bias set to 0 ($b = 0$). We notice that the dip occurs around the region where the gap between p_c and p_w is smallest.

Another way of explaining this behavior is that EB will perform well whenever instances of different classes follow widely different distributions on the possible labels that can be predicted for them. The exact distribution is not important, in so far as they differ widely from one class to another.

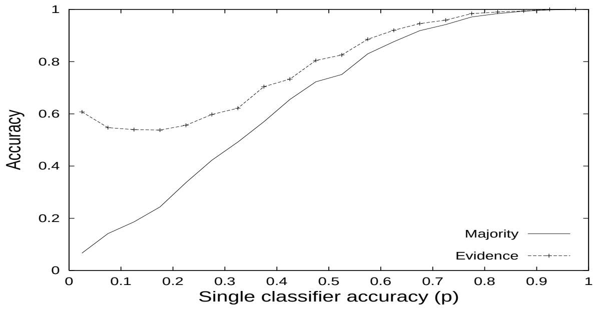

Figure 1. Changing single classifier accuracy

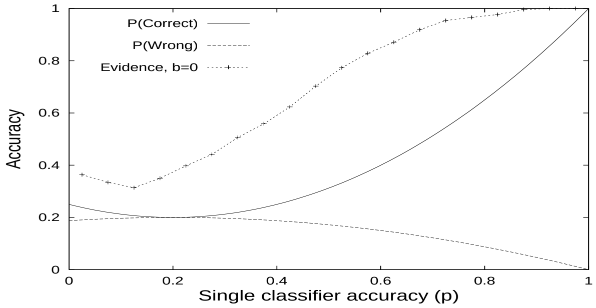

Figure 2. Estimated probabilities of evidences

Number of Models Figure 3 shows that as the number of models (n) increases from 2 to 50, both methods approach close to 100% accuracy. However, when we repeat these experiments for a smaller value of p (0.2), accuracy of MV drops with increasing n whereas EB keeps improving as shown in Figure 3.

Majority is known (Breiman, 1996) to have this bi-modal behaviour where below a certain value of p increasing n causes overall accuracy to decrease whereas above that it consistently improves. For $k = 2$ the turn over point is $p = 0.5$ and for general k it is $\frac{1}{k}$.

We next attempt to understand why even for smaller

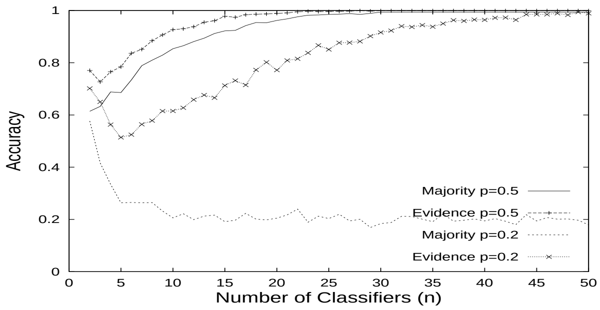

Figure 3. Number of Models

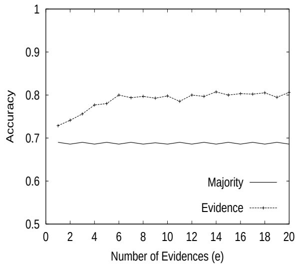

Figure 4. # Evidence (e)

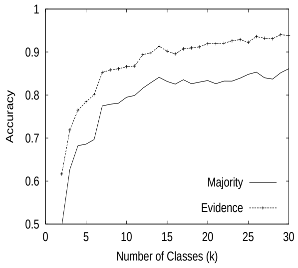

Figure 5. # Classes (k)

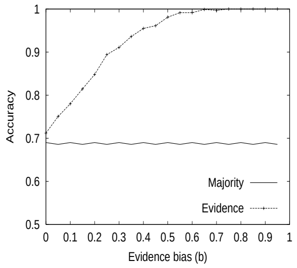

Figure 6. Evidence bias (b)

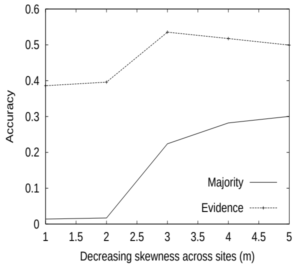

Figure 7. Site bias (m)

p the accuracy of EB does not drop with increasing n. Consider the evidences E_c and E_w from the correct and wrong sites again. The proximity of each evidence is the sum of matches over all sites. As shown in Figure 2, the match probability p_c of E_c is higher than the match probability p_w of E_w. The expected match over $n-2$ sites for each of these evidences is $(n-2)p_c$ and $(n-2)p_w$ respectively. As long as p_c is greater than p_w, increasing n causes the total proximity of E_c to be increasingly larger than E_w's.

The slight drop in accuracy of EB for small values of n (Figure 3) is because in case of ties we randomly choose one of the contending class. This causes a larger fractional increase in accuracy when n is small and when the probability of the correct class p is close to the probability of the incorrect class. Lack of space prevents us from dwelling on this issue further.

Number of Evidences As expected accuracy of EB improves with increasing number of evidences as shown in Figure 4. Even with one evidence, EB boosts accuracy by 3% above MV. Interestingly, whereas increasing number of models n causes accuracy to touch 100% in the limit, increasing evidences does not boost accuracy beyond a certain point. The reason is that whereas sites are independent of each other, evidences are not. Evidence proximities are correlated through the common prediction vector of T that restricts what proximities the evidences get assigned. For example, with the settings of Table 1, if T with true class C_5 gets its prediction as $(C_5, C_3, C_3, C_3, C_3)$ then for our settings, it is unlikely for site 1's evidence to win in spite of its bias (b) of 0.1. A simple envelope calculation will show that the match probability of site 1 is 0.2 whereas the wrong sites is 0.38. Although such a prediction vector has low probability, when it does occur, increasing evidences cannot save the situation.

Number of classes Keeping single model accuracy fixed, increasing number of classes causes both algorithms to perform better. Majority improves because fewer number of models now need to be correct. The chances of any single wrong class winning becomes smaller because the value $\frac{1-p}{k-1}$ (the probability of a site predicting a given wrong class) decreases with increasing k. The reason why EB improves is found by comparing the expressions for the match probability of the wrong site p_w and right site p_c in Equations 2 and 1 respectively. We notice that p_w decreases faster with k than p_c. Another way to explain the improvement of EB is that increasing k causes the gap between the class distribution of the wrong and correct classes to increase.

Effect of Evidence bias As expected increasing the proximity of the evidence to the test instance will cause accuracy to increase (Figure 6). The interesting point is that even when the bias is zero, EB is slightly better than majority.

Effect of site skewness EB is designed to perform well when different sites are experts at different classes. We model skewness using a parameter m such that m sites will correctly predict a class with probability p/m and the remaining $n-m$ sites will make the wrong prediction. For each class we choose its set of m specialist sites randomly. For these experiments $p = 1$. Thus when $m = 2$, 2 sites predict the correct class with probability 0.5 and 3 sites predict one of the wrong classes. When $m = 5$, all sites predict the correct class with a probability of 0.2. From the graphs in Figure 7 we notice that when $m = 1$ and 2, that is, when exactly one or two sites give the correct prediction the accuracy of MV drops to zero whereas EB remains within 10% of the uniform case.

Summary EB is better than MV for a wide range of settings. The gap is larger when

- the predicted class distribution of instances from different class are widely separated as we observed when p was made either too large or too small or k was made large,
- the number of sites is increased, reaching 100% accuracy in the limit irrespective of p,
- the number of evidences is large but benefits of

increasing evidences peaks off at a value less than 100%,

- evidences of correct site are close to the test instance,
- classifiers are specialists in different classes.

5. Real datasets

We present experiments emulating the document classification scenario described in Section 1. We use the Naive-bayes classifiers (multinomial model) to classify documents. We considered two datasets. First is the **Twenty-newsgroups datasets** available from the UC Irvine KDD repository (Bay, 1999). This dataset consisted of 20,000 newsfeed articles divided equally amongst 20 Usenet newsgroups. Each class thus had a total of 1000 documents. Second is the **ODP's Recreation dataset** consisting of web pages collected from nine sub-categories under the Recreation sub-tree of a popular web directory (Open Directory Project[5]). There were a total of 10039 documents and the number of documents per class ranged from 490 to 1933.

In all experiments the training and test data were obtained after a standard 70-30 split. Training data was disjoint partitioned equally and randomly across all sites.

In Tables 2 and 3 we compare the accuracy of MV and EB with changing number of sites (n) for both datasets (labeled Web and News). While changing n the amount of data per site was kept constant. For each of the datasets, we achieved this by initially equi-partitioning all available training data into the maximum number of sites. For each n, we then picked random subsets of n sites. For the Web dataset, as the number of sites is increased from 2 to 15, accuracy of EB improves from 78.7% to 84.4% and majority from 76.4% to 80.6%. The gap between EB and MV increases from 2% to 4% at 15 sites. For the newsgroup dataset, as the number of sites is increased from 2 to 10, accuracy of EB improves from 49.1% to 63.4% and majority from 44.3% to 57.2%. EB is about 6.2% better than MV at ten sites.

In Table 4 we show accuracy with increasing number of evidences (e) for the Web data partitioned into five site (n=5) and News data partitioned into three sites (n=3). The amount of data per site is significantly larger than for the previous experiments with increasing n. As the number of evidences increases from one to five, the News data shows a boost in accuracy by 1.5% and Web data by 0.5%.

[5]`http://dmoz.org` - The classes are *Amateur Radio, Birdwatching, Boating, Guns, Kites, Living History, Models, Roads and Highways, Scouting*

n	2	3	5	7	10	12	15
MV	76.4	78.3	79.1	79.7	80.5	81.9	80.6
EB	78.7	80.2	82.1	82.6	83.7	84.5	84.4

Table 2. Web: Varying number of sites (5 evidences)

n	2	3	5	7	10
MV	44.3	46.8	52.1	55.5	57.2
EB	49.1	51.5	56.7	61	63.4

Table 3. News: varying number of sites n (5 evidences)

e	1	2	5	7	10
Web, n=5	85.1	85.3	85.6	85.5	85.5
News, n=3	75.3	75.9	76.8	76.3	76.8

Table 4. Varying number of evidences

These experiments show that EB has promise of performing well in real-life settings. More experiments with skewed partitioned datasets and hybrid classifications methods are ongoing.

6. Discussion

The Evidence-Boosting algorithm provides a novel bridge from the previous model-dependent notions of proximity discussed in Section 2 to a model-neutral notion of relevance using an interesting two-phase approach.

Previous methods relied on a per-site confidence score that represented coverage and/or performance around the test instance. These scores are often not comparable across classification methods. Even for the same classifier type, the differences in the final model structure (for example, different shapes of decision trees or different distance functions in k-NNs) make the score incomparable across sites. However, locally within a site, it is meaningful to compare instances based on these scores. We exploit this insight in first locally choosing the e closest evidences using very similar model-specific scoring techniques.

In the next phase, we compare relevance across models relying on a model-neutral property of classifiers that two instances from the same class are more likely than instances from different classes to have the same prediction. While this property may not hold strongly for two arbitrary instances, this is especially true for evidence and test instance pairs because local classifiers already choose them to be close to each other. We rely on no other property of a classifier: in particular even though we rely on internal proximity measures in choosing evidences, we do not assume that predictions are primarily influenced by instances near the test case. The method is thus applicable to a large class of the models including those whose predictions depend on on global characteristics of the data - for example, Bayesian classifiers and logistic regression.

Generalizations, Limitations and Extensions In this paper, each site's prediction consisted of a single class. The general case where classifiers output a ranked list of classes or a score for each class can be handled in the same framework. We just need to generalize the proximity function to take as argument two $n \times k$ matrices of scores instead of n-dimensional vectors of class predictions. In addition, when some individual classifiers do volunteer information like overall predictive performance, we need to redesign our weighting functions to take these into account.

A limitation of this method is that classifiers have to be now extended to return evidences. Although, the proposed evidence selection method is streamlined with the training task, extra work is needed in clustering and retaining the candidate evidences. During the classification process evidences impose communication and processing overheads.

The basic idea of dynamically probing individual classifiers at the time of prediction can be extended further. For instance, instead of the two phase interaction used in Evidence-Boosting, could we do better through multiple rounds of responses and interactions amongst classifiers? Would it help if a site gives evidences of the opposite class? If every site knows of every other site's prediction on the instance, can it do anything different? We hope this paper will trigger research in all these areas to make ad hoc sharing of mining models a reality.

7. Conclusion

We presented a new framework for impromptu-selection of models using dynamically generated evidences to establish the expertise of a site around the test instance. The proposed algorithm builds on prior work on dynamic model selection and extends them so as to not require prior meta-learning and be model-neutral. Analytical studies and experiments on real-life datasets show that the algorithm is superior to majority voting for a wide range of settings obtained by changing classifier accuracy, number of sites and classifiers and relative expertise of different sites.

The paper was a first attempt at providing a framework for evidence-based impromptu model selection. There are several variations of this basic framework involving myriad ways of interacting with classifiers when lazily learning to combine them. We hope this paper will encourage further research in designing such models that enable them to be shared by people who may not have any knowledge of its training process.

References

Bauer, E., & Kohavi, R. (1996). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning Journal*, *36*.

Bay, S. D. (1999). The uci kdd archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

Breiman, L. (1996). Baggin predictors. *Machine Learning*, *24*, 123–40.

Chan, P., & Stolfo, S. (1995). A comparative evaluation of voting and meta-learning on partitioned data. *In Proceedings of the Twelvth International Conference on Machine Learning ICML-95*.

Dietterich, T. G. (1997). Machine-learning research. *AI Magazine*, *18*, 97–136.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*, 139–157.

Koppel, M., & Engelson, S. (1996). Integrating multiple classifiers by finding their areas of expertise. *AAAI-96 Workshop On Integrating Multiple Learned Models.*.

Merz, C. (1998). Using correspondence analysis to combine classifiers. *Machine Learning*.

Pennock, D. M., II, P. M.-R., Giles, C. L., & Horvitz, E. (2000). A normative examination of ensemble learning algorithms. *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)* (pp. 735–742).

Prodromidis, A., Chan, P., & Stolfo, S. (1997). Meta-learning in distributed data mining systems: Issues and approaches. *In Proc. of the 3rd Int'l Conf. on Knowledge Discovery and Data Mining*.

Sarawagi, S., & Nagaralu, S. (2000). Data mining as an internet service. *Sigkdd Explorations*, *2*.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*, 297–336.

Tresp, V., & Taniguchi, M. (1995). Combining estimators using non-constant weighting functions.

Tsymbal, A. (2000). Decision committee learning with dynamic integration of classifiers. *Current Issues in Databases and Information Systems, Proceedings of ADBIS-DASFAA 2000*.

Tsymbal, A., Puuronen, S., & Terziyan, V. (1998). Advanced dynamic selection of diagnostic methods. *Proceedings of the Eleventh IEEE Symposium on Computer-Based Medical Systems* (pp. 50–54). IEEE CS Press.

Webb, G. I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, *40*, 159–196.

Weigend, A. S., & Nix, D. A. (1994). Predictions with confidence intervals (local error bars). *ICONIP, Seoul*.