Separating Prediction from Reliability Prediction

Anonymous Author(s)
Affiliation
Address
email

Abstract

Estimating the reliability of predictions generated by a classification algorithm is of great practical importance. Existing techniques typically use by products of the classification process in order to address this task. For example, for linear classifiers, the margin of a test instance from the closest hyper-plane often serves as a proxy for prediction reliability. Here, we propose a different strategy, in which we treat reliability estimation as a separate learning task. Thus, given the predictions of the classifier over some training data, we construct a new classifier that aims to identify instances for which the predictions generated by the original classifier are unreliable. The proposed approach is general and can be used on top of any classification procedure. We demonstrate its performance while using the classical Naive Bayes classifier. Experimental results over several benchmark datasets, demonstrate the potential advantages of the proposed method.

1 Introduction

In the classical supervised learning paradigm, the classifier is provided with labelled training instances and its goal is to predict the class label of new, previously unseen, test instances [1]. Naturally, for different test instances, a typical classifier will generate predictions at different levels of reliability. Estimating the reliability associated with each prediction could be important in various scenarios. For example, in the active learning paradigm the classifier typically requests the true label only for relatively unreliable predictions [2]. Thus, proper estimation of prediction reliability is essential for the success of active learning techniques. In addition, in different practical applications, identifying unreliable predictions may assist the user to avoid wrong decisions.

Estimating prediction reliability is typically done via straightforward techniques. For linear classifiers, where the learning rule is represented by hyper-planes, the margin of a test instance from the closest hyper-plane can serve as a proxy for prediction reliability, where larger margins imply stronger reliability [3]. Alternatively, some classifiers work by assigning a score per possible label; then, the gap between the top score and the second best score, can serve as an estimate of prediction reliability where larger gaps presumably imply more reliable predictions [7, 4]. Some learning algorithms, like the Naive Bayes (NB) classifier [5], aim to directly estimate the class membership probability associated with a test instance. Here, obviously the properties of the estimated class membership probability can be used to approximate prediction reliability, where relatively uniform class distributions are expected to be associated with less reliable predictions [6, 7].

What is common to these techniques is that reliability is estimated by exploiting some innate properties of the classifier, i.e., via considering by products of the classification process itself. Here, we propose a different strategy, in which we *treat reliability estimation as a separate learning task*. Thus, given the predictions of the classifier over some training data, we construct a new classifier, termed henceforth the Reliability Classifier (RC), that aims to distinguish instances for which the predictions generated by the original classifier were reliable, from instances for which the generated predictions were not reliable. Our underlying premise is that often there exist a statistical signature that distinguish these two types of instances. The RC aims to exploit this statistical signature in order to directly asses whether a particular prediction should be considered reliable or not.

Applying the RC naturally results with an alternative view of the training data. E.g., data features that are irrelevant for the original classification task could be useful for the RC. Further, the RC can use additional features that were not used within the original classification task. In particular, any by products of the original classification process that are used by common techniques for estimating prediction reliability, can be exploited as additional features by the RC.

Our approach is general and can be used on top of any classification method. Further, there is no limitation on using one type of classifier for the original classification task, accompanied by a different classification scheme employed as the RC. Here, we demonstrate the performance of our approach while using the classical NB classifier [5] for both classification tasks. Experimental results over benchmark datasets demonstrate the potential advantages of the proposed method.

2 Causes for unreliable predictions

Before describing our approach it is useful to discuss possible causes of unreliable predictions. *Insufficient Training Data*: Given smaller training sets, one should expect less reliable predictions. Notice, however, that the impact could change between different instances. Consider, for example, the problem of text classification with relatively small training data. Here, the reliability of predicting the topic of a document that includes mainly frequent words will typically be less affected, compared to a document that includes mainly infrequent words. *Non Valid Modeling Assumptions*: The constructed classifier often involves non valid modelling assumptions that may yield unreliable predictions for some instances, as we discuss in more detail in Section 7.2. *Inherent uncertainty*: Some uncertainty is inherent in the problem setting and cannot be alleviated by increasing training set size or using stronger learning techniques. For example, attempting to predict the outcomes of a stochastic process is inherently uncertain. Still, even in this case the reliability may vary between different regions in the feature space. Clearly, unreliable predictions could arise from any subset of these three possible causes. In addition, each cause may have different implications and may require a different treatment, as we discuss below.

3 The Reliability Classifier

Let $\mathbf{X}$ be an m-dimensional feature space, Y be a finite set of classes, and $P_D(\mathbf{X}, \mathbf{Y})$ be a distribution over $\mathbf{X} \times \mathbf{Y}$. We assume that we are given training data that consist of N instances, sampled from P_D, denoted as $\{(\mathbf{x^{(1)}}, \mathbf{y_1})\}_{\mathbf{i=1}}^{\mathbf{N}}$, where $\mathbf{x^{(i)}} \in \mathbf{X}$, $\mathbf{y_i} \in \mathbf{Y}$, and $\mathbf{x_j^{(i)}}$ indicates the measurement observed for the i-th instance with respect to the j-th feature. Next, we denote our classifier by h, which is a function from $\mathbf{X}$ to Y. Given an instance, $\mathbf{x}$, we denote by $h(\mathbf{x})$ the class label predicted for it by h. Our goal is to develop a scheme that amongst all predictions generated by h, allows to distinguish the reliable predictions from the non-reliable ones. Loosely, it seems reasonable to require that predictions that are defined as "reliable" will tend to be more accurate than predictions defined as "unreliable". However, this does *not* mean that predictions that are defined as "unreliable" should be always, or even most of the time, incorrect. As a simple example, let us consider a binary classification task where the two classes are equally frequent. Next, let us assume that within some regions in feature space, $\mathbf{X}$ and Y are essentially independent. Thus, it is plausible that for instances located within these regions, h will predict the class label completely at random. Therefore, the predictions for these instances should be considered "unreliable", although about half of these predictions will be correct.

Given the goal stated above, the general strategy proposed in this work is to train an additional classifier which is dedicated to distinguish the reliable predictions of h from the non reliable ones. We term this classifier the *Reliability Classifier* (RC) and denote it via R_h, to emphasize its relation with the original classifier, h. Thus, given an instance $\mathbf{x}$, R_h will aim to determine whether the prediction $h(\mathbf{x})$ should be considered reliable or not. Formally, R_h could be thought of as a function from $\mathbf{X}$ to $\{0, 1\}$, where $R_h(\mathbf{x}) = \mathbf{1}$ implies that $h(\mathbf{x})$ should be considered "reliable" and $R_h(\mathbf{x}) = \mathbf{0}$ implies that $h(\mathbf{x})$ should be considered "unreliable". In the following section we discuss possible routes to construct and train R_h.

4 Training the Reliability Classifier

In order to train the RC one must define its training data, i.e., to define the training instances and their associated "reliability" labels, as we discuss next.

4.1 Defining the reliability training examples

The proposed strategy implies that now one should exploit the N training instances for training *two* classifiers, h and R_h, each focused at a different goal. One possibility, that aims to avoid overfit issues, is to split the training data and use $f \cdot N$ samples as training data for h, while using the remaining $(1 - f) \cdot N$ samples in training R_h. We will refer to this alternative as *Split Training*. Notice, that under this mode, a natural tradeoff arises; as f decreases one often expects a reduction in the accuracy of h predictions, that may turn the task of R_h easier. At the extreme, as $f \rightarrow 0$, h essentially predicts the class labels arbitrarily, hence R_h can simply declare all h predictions as "unreliable". In Section 7 we explore this trade-off. However, if N is relatively large, it might be that there is a wide range of f values for which each of the two classifiers receives sufficient training data to attain good performance.

An alternative approach is to use all N training instances twice; once for training h, and again for training R_h. We will refer to this mode as *Complete Training*. Notice that both classifiers are aiming at very different goals, similarly to text classification algorithms that can be trained over the same training data, to determine the topic or the author of a new incoming document [8]. Our results in Section 7 demonstrate that this mode typically yields good performance, hence we mainly report results under this *Complete Training* mode.

4.2 Defining the reliability labels

Once the training instances for R_h have been determined, one must define their associated "unreliable" and "reliable" labels. Obviously, this definition will direct the performance of R_h. Apparently, there are different ways by which this issue could be addressed within our framework. Several alternatives are described next, under the Complete Training mode. There are certainly other legitimate alternatives, that can be explored in future work.

- *Stability*. Here, one randomly splits the training data in two halves, trains h using one half of the instances and uses the obtained classifier to predict the class labels for the remaining half. Repeating this process $2n$ times, one can obtain an average of n predictions of h per instance. The relative stability of these predictions – as measured by some common statistic, e.g., Shannon's Entropy [9], can be used along with a pre-defined threshold to associate each instance with an appropriate reliability label.
- *Correct Fraction*. This scheme is similar to the Stability scheme, but instead of utilizing the relative stability of the obtained predictions, one considers the fraction of correct predictions obtained per instance. This fraction, along with some pre-defined threshold, can be used to determine the reliability label per instance.
- *Correct vs. Incorrect*. Here, one trains h over all N training instances and then apply it to predict the label for each of these instances. Instances for which the obtained prediction was correct ($h(\mathbf{x^{(i)}}) = \mathbf{y_i}$) will be labelled as "reliable", while instances for which the obtained prediction was incorrect ($h(\mathbf{x^{(i)}}) \neq \mathbf{y_i}$) will be labelled as "unreliable".

In general it seems that the first two schemes will be more appropriate in scenarios where the unreliability arises due to insufficient training data accompanied by some inherent uncertainty in P_D (see Section 2). In the current work we use the third – *Correct vs. Incorrect* scheme – mainly since it is relatively simple, less computationally intensive, completely non-parametric, and in particular involves no threshold definition. However, as mentioned, we note that this labelling scheme only approximates the general notion of "reliability" that we aim to capture. In particular, it is expected that some instances – that were classified correctly by h merely by chance – will be unjustifiably labelled as "reliable". Nonetheless, our results in Section 7 demonstrate that even under this approximated labelling scheme, our proposed approach outperforms standard benchmark techniques.

4.3 Exploiting additional features

Clearly, features that were found irrelevant by h, may still be informative and useful in the context of the classification task performed by R_h. However, even beyond that, we note that our proposed framework naturally allows to exploit *additional* features by R_h. In many cases it may be advantageous to add features describing meta-data properties of the training data, such as document length for text classification tasks, or percentage of missing data for incomplete datasets. Furthermore, any property used by conventional techniques to estimate the reliability of h predictions can be added

Input

N labeled training instances, $\{(\mathbf{x^{(i)}}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\mathbf{N}},\ \mathbf{x_i} \in \mathbf{X},\ \mathbf{y_i} \in \mathbf{Y}$
Classifier $h : \mathbf{X} \rightarrow \mathbf{Y}$ and Reliability Classifier $R_h^* : \mathbf{X} \rightarrow \{\mathbf{0}, \mathbf{1}\}$

Training

Train h using the N training instances
Apply h over all N training instances
Define $r_i = 1$ iff $h(\mathbf{x_i}) = \mathbf{y_i}$ and $r_i = 0$ otherwise
Train R_h using $\{(\mathbf{x^{(i)}}, \mathbf{r_i})\}_{\mathbf{i=1}}^{\mathbf{N}},\ \mathbf{x_i} \in \mathbf{X},\ \mathbf{r_i} \in \{\mathbf{0}, \mathbf{1}\}$

Testing

Given a new, previously unseen, test instance, $\mathbf{x}' \in \mathbf{X}$:
$h(\mathbf{x}) \in \mathbf{Y}$ is the predicted class label for $\mathbf{x}$
$R_h(\mathbf{x}) \in \{\mathbf{0}, \mathbf{1}\}$ is the predicted reliability for $h(\mathbf{x})$

Figure 1: Pseudo-code for the proposed approach, using *Complete Training* mode, and the *Correct vs. Incorrect* labelling scheme. In our experiments, we used the standard NB classifier for both h and R_h. Training R_h^* is done similarly to training R_h, except that the MAP and Entropy of h are added as features to $\mathbf{X}$

as a *feature* in the representation of instances considered by R_h. For example, if h is a separating hyperplane, the margin from the hyperplane can be used as additional input for R_h. If h is a Naive Bayes classifier. it generates an estimate of the entire class membership distribution, $P(y|\mathbf{x})$. Various statistics extracted from this distribution can be used as additional features considered by R_h, such as the MAP value – $P(y^*|\mathbf{x}),\ \mathbf{y}^* = \mathbf{argmax_y P(y|x)}$.

5 Algorithm

In Fig. 1 we provide Pseudo-code that summarizes the main steps in the proposed approach, under the *Complete Training* mode, and while using the *Correct vs. Incorrect* scheme to define the reliability labels. In our experiments we used the classical NB classifier [5] both as h and as R_h, mainly due to its simplicity, and due to the fact it produces by products that can serve conventional benchmark methods for estimating predictions' reliability. We emphasize that there is no limitation in using different classification schemes for h and R_h. Nonetheless, for brevity purposes, here we used the same classification scheme for both tasks. For the original classification task, the NB classifier works by estimating $P(X_j|Y),\ \forall j = 1 : m$ based on the labelled training instances. Next, using the NB class-conditional independence assumption, given a new instance, $\mathbf{x}' \in \mathbf{X}$, the class membership a-posteriori distribution is estimated via

$$P(Y|\mathbf{x}') \propto \mathbf{P(Y)\Pi_{j=1}^{m} P(X_j = x_j'|Y)}\ , \tag{1}$$

and $h(\mathbf{x}')$ is defined as $argmax_{y \in Y} P(Y|\mathbf{x}')$. For the RC, R_h, the same procedure is applied while using the reliability labels instead of the class labels, Y. In addition, we tested another version of R_h, denoted R_h^*, that uses two additional features for representing $\mathbf{x}'$, that are extracted from $P(Y|\mathbf{x}')$ – the MAP value and Shannon's Entropy [9], defined as $-\sum_y P(y|\mathbf{x}') \log \mathbf{P}(\mathbf{y}|\mathbf{x}')$.

6 Experimental Details

6.1 Baseline algorithms

We compare our approach to three known methods to estimate reliability, which exploit the posterior distribution of class labels generated by the NB Classifier. The common concept behind these methods is that when the posterior distribution tends to be more uniform the prediction of the NB classifier is presumably less reliable. We note that all these methods will mainly capture unreliability arising from uncertainty inherent to the data distribution.

- MAP – Measures the posterior probability of the most probable class.
- Entropy – Measures the Shannon Entropy [9] of the posterior distribution [6, 7].
- Gap – Measures the difference between MAP value and second most probable class [7, 4].

In classification tasks comprised of two classes these methods will give equivalent results. With more than two classes, each method may perform differently.

6.2 Evaluation

As discussed in Section 3 it seems reasonable to require that predictions that are identified as "reliable" will be on average more accurate then predictions identified as "unreliable". To quantify the level by which we achieve this goal we use ROC curves, where we define as "positives" the instances correctly predicted by h, and as "negatives" the incorrect predictions. Examining different thresholds for declaring a prediction as "reliable" creates different rates of true positives (correct predictions declared "reliable") and false positives (incorrect predictions declared "reliable"). To quantify the performance of each method we use the area under the curve (AUC) [10]. Finally, for completeness we further report the performance of h in the original classification task, in terms of Macro-averaged Precision/Recall [11].

6.3 Datasets

We compare our method to benchmark methods for a synthetic example (see below), 5 popular datasets from the UCI repository [12], and one clinical dataset [13] as described next (cf. Table 1). *Abalone* – samples of Abalones described by physical features; target: number of rings on the abalone shell [12], that varies from 1 to 29; we reduced the number of classes to 3 by quantizing classes into 3 equally populated bins. *Cardiotocography* – automatically processed cardiotocograms; target: fetal state [12]. *Contraceptive Method Choice (CMC)* – surveys of women about demographic and socio-economic characteristics; target: contraceptive method used by the woman [12]. *Mushroom* – hypothetical samples of mushrooms corresponding to 23 species, described by physical features. The data contains missing values in one categorical feature that we represented as an additional category for that feature; target: mushroom class – edible or poisonous [12]. *Wine Quality* – red wine samples described by physicochemical test; target: wine quality [12]. *EuResist* – contain data for 48,625 HIV patients. We used a subset, describing 6,056 treatments, for which a bigger set of features is available. For each treatment we considered: viral load before treatment; number of past treatment lines; Amino-Acid sequence of two viral proteins – Protease and Reverse Transcriptase; and the administered drugs. The class label is a binary feature indicating success in reduction of viral load following the treatment [13]. Each treatment was represented by 24 binary variables, one per different drug included in these data, indicating whether it was used within this particular treatment. The two viral protein sequences were represented by 99 and 440 categorical features, respectively, each with 23 possible values – 20 represent the 20 different amino-acids, and the remaining 3 represent stop codon, deletion and unknown amino acid [13].

6.4 Run details

For each dataset, all instances were randomly divided to equally sized train and test sets. This process was repeated 10 times, and for each algorithm the average results and the associated standard deviation are reported. Benchmark methods used all the training data to train the NB classifier, that was then used to classify all instances in the test set. The benchmark methods then used the obtained class posterior distribution to estimate predictions' reliability, as explained above. For the Reliability (NB) Classifier we tried both *Split Training* with $f = 0.1 \ldots 0.9$ and *Complete Training* (see Fig. 1). The reliability labels were defined via the *Correct vs. Incorrect* scheme. In each case we tested two versions: the first using the original representation of the data (R_h); and the second that further used the MAP and Entropy values per instance, as explained above (R^*_h). All classification tasks were performed using Matlab's NaiveBayes classifier.

7 Experimental Results

7.1 *Split Training* vs. *Complete Training*

For all datasets, *Complete Training* led to better or comparable results to *Split Training*. Specifically, as expected, for small f values, the quality of h predictions decreased. As f was increased, the quality of h's predictions increased as well (Fig. 7.1 b, c, e, f), but at the cost of decreased performance of R_h (Fig. 7.1 a, d). Although for some datasets we identified ranges of f values which results in good performance of R_h with negligible impact on h (e.g., $f = 0.5 \ldots 0.9$ for the EuResist dataset – Fig. 7.1 top), even the best performance of R_h under *Split Training* were not superior to the R_h performance under *Complete Training*. In addition, *Complete Training* avoids the need of tuning f. Thus, henceforth we report results only for this *Complete Training* mode.

Table 1: **Datasets Description.** Description of the datasets used in the experiments.

Name	Instances	Features	Feature types	Classes	Missing Values
Abalone	4177	8	categorical + real	3	no
Cardiotocography	2126	23	real	3	no
CMC	1473	9	categorical + real	3	no
EuResist	6056	85	categorical	2	yes
Mushroom	8124	22	categorical	2	yes
WineQuality	4898	12	real	11	no
Syn1	2000	2	categorical + real	2	no

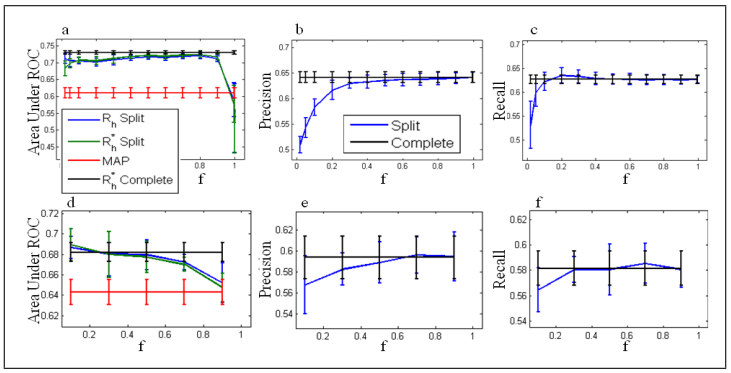


Figure 2: **Performance of h and R_h depends on the division of training data.** Area under ROC curve (left), Macro averaged Precision (middle) and Recall (right) of h as a function of f – the fraction of training data used to train h (x-axis), for EuResist dataset (top) and Abalone dataset (bottom).

7.2 Synthetic Example

Syn1 - is a synthetic dataset generated for this study to demonstrate the performance of our method when unreliability of predictions arises from invalid model assumptions and inherent uncertainty in the data distribution. Our synthetic example aims to imitate a situation in which one feature (X_1) is highly correlated with the class label. However, for some instances this feature is very noisy, thus breaking the correlation for those instances. A second feature in the data (X_2), indicating the amount of noise in the first feature, X_1. For example, let the instances represent patients, the class label sick/healthy, and X_1 a blood test which is highly indicative of the disease. Further, we assume that blood samples of different patients were analysed in two different laboratories, one of which was contaminated, inserting noise into the blood test results, and the identity of the lab is represented by X_2. Obviously, predictions for patients tested in the contaminated lab are less reliable. The dataset was generated in the following manner: X_1 and class label where sampled from the multivariate normal distribution with $\mu = 0$, $\Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}$ The label was then converted into a binary feature by setting all values smaller than 0 to class y_1 and all remaining values to class y_2. X_2 was sampled from the Bernoulli distribution with $p = 0.6$. For all instances in which $X_2 = 1$ the value of X_1 was replaced with a sample from the normal distribution with $\mu = 0$ $\sigma = 1$. In this dataset, the unreliability is inherent to the data. Furthermore, the modelling assumptions of the NB classifier are invalid in this situation, as it assumes all features are independent given the class label. Thus, for methods that rely on the posterior distribution of classes to model unreliability, it will be harder to capture the unreliability in this situation. Indeed, comparing the AUC of R_h and benchmark methods for this example (Fig. 7.3 a), depicts clear advantage for R_h.

7.3 Reliability estimation results

The performance of all methods for all datasets in terms of estimating predictions reliability are presented in Table 2. For completeness we also report the results of the NB classifier, h, for the original classification task (Table 2).

Evidently, for all datasets, R_h outperforms the benchmark methods. For example, for the EuResist dataset, the difference between the AUC of R_h and benchmark methods is 9 standard deviations (Table 2, Fig. 7.3 b). I.e., estimating reliability directly from the class a-posteriori distribution, estimated by the NB classifier, as done by the benchmark techniques, yield inferior performance. This implies that this a-posteriori distribution is not properly estimated, i.e., that the sources of unreliable predictions in these datasets can not be solely attributed to inherent uncertainty in P_D, and must include non-valid modelling assumptions and/or insufficient training data. Finally, we note that R_h^* that further used the MAP and Entropy features, obtained better results than R_h, although often this improvement seems negligible.

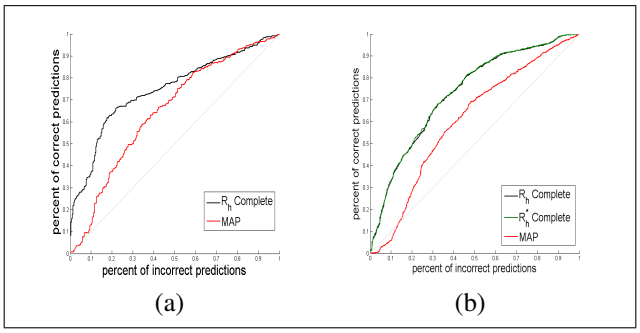


Figure 3: **ROC curve**. (a) Percent of Syn1 test instances correctly predicted by h out of all correct prediction (x-axis) vs. percent of incorrect prediction out of all incorrect prediction (y-axis) when considering only predictions that received a reliability score above a threshold, for varying thresholds. (b) Same as (a) for the EuResist dataset.

7.4 Different features are informative for each classification task

To explore how informative is each feature X_j in each prediction task, for each task we examine the Jensen-Shannon divergence [14] between the classes; $JS_h \equiv JS(P(y_1|X_j), P(y_2|X_j))$ indicates how well X_j distinguish between the two classes in the original classification task, while $JS_{R_h} \equiv JS(P(r_1|X_j), P(r_2|X_j))$ indicates how well X_j distinguish the "reliable" predictions of h from the "non-reliable" ones. In the EuResist dataset the most significant differences in these JS values were found in the distribution of Amino Acids along different positions of the Reverse Transcriptase protein. While for the majority of positions, the JS is very similar, there are some substantial deviations from this trend (Fig. 7.4 a). Specifically, for positions in which the probability of known and unknown amino acids are relatively similar, JS_{R_h} was typically higher than JS_h. Further examinations of these positions reveal that unknown amino acids are more common in the "unreliable" class, matching the intuition that predictions are less reliable as missing data rate increases. Another example arises from examining these JS values for features in the Mushroom dataset. We observe that mushroom's cap color is not very informative for distinguishing poisonous mushrooms from edible ones, as the distribution of color is similar in both classes (Fig. 7.4 b, top). However, this feature can help distinguish reliable from unreliable predictions (Fig. 7.4 b, bottom); e.g., it seems that predictions for white mushrooms (denoted by 'w') tends to be unreliable.

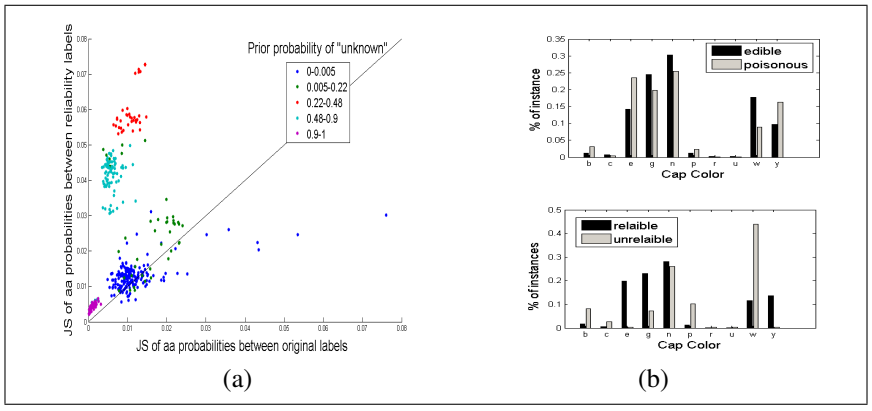
8 Discussion

We presented a framework to estimate classification reliability by modelling it as a learning task separated from the original classification task. Experiments on several datasets show that our method outperforms commonly used benchmark techniques.

A powerful approach for estimating uncertainty is the Bayesian approach. This approach assumes data is generated by some probabilistic model, whose unknown parameters themselves are assumed to be distributed according to a particular distribution, e.g., Dirichlet distribution. Given a training set, it is then theoretically possible to compute the posterior distribution of the model parameters, which allows to calculate the posterior class probability for a given test instance. From this, confidence intervals may be calculated directly or approximated [15]. This approach accurately models uncertainty stemming from training data size, as well as the inherent uncertainty. However it does

Table 2: **Comparison of Reliability Estimation.** Performance of all reliability estimation methods, measured using the area under the ROC curve. Performance of NB Classifier for the original task is reported in terms of Macro average Recall, and Precision. For each dataset we report the mean area (first row) and the standard deviation (second row) calculated based on 10 repeats.

Dataset	R_h	R_h^*	MAP	Ent	Gap	Precision	Recall
Abalone	**0.683**	0.682	0.643	0.651	0.638	0.593	0.581
	0.009	0.009	0.012	0.008	0.013	0.021	0.014
Cardiotocography	0.804	**0.844**	0.769	0.769	0.768	0.767	0.761
	0.020	0.018	0.035	0.035	0.035	0.026	0.021
CMC	0.657	**0.670**	0.646	0.642	0.642	0.512	0.513
	0.025	0.025	0.024	0.022	0.025	0.016	0.015
EuResist	0.724	**0.727**	0.607	0.607	0.607	0.628	0.643
	0.007	0.007	0.013	0.013	0.013	0.007	0.009
Mushroom	0.987	**0.992**	0.969	0.969	0.969	0.954	0.947
	0.004	0.002	0.001	0.001	0.001	0.003	0.004
WineQuality	0.599	**0.602**	0.566	0.552	0.567	0.283	0.305
	0.011	0.014	0.011	0.011	0.010	0.031	0.013
Syn1	**0.777**	0.737	0.630	0.630	0.630	0.725	0.724
	0.010	0.015	0.012	0.012	0.012	0.009	0.009



(a) (b)

Figure 4: **Features Contribution Depends on Classification Task.** (a) The JS of Amino Acids probabilities for each position in the reverse Transcriptase protein (coloured dots) between the two original classes (x-axis) and the two reliability classes (y-axis). The fraction of instances in the training data for which the amino acid in this position is unknown is indicated by the dots color. (b) Distribution of mushroom's cap color for edible (black bars) vs. poisonous (grey bars) mushrooms (top) and for reliable (black bars) vs. unreliable (grey bars) predictions (bottom).

not address uncertainty resulting from an erroneous model. Moreover, it requires an additional assumption of a prior distribution, which is usually unknown. Finally, calculations involved in this approach are usually intractable [15] and require approximation algorithms.

The datasets used here are relatively large and perhaps less appropriate for testing unreliability arising from limited training data. Future work can more directly explore the performance of our approach in such scenarios, which in particular may require a different labelling scheme, more in the spirit of the *Correct Fraction* method described in Section 4.2.

Here, we considered reliability estimation as a binary classification task, aiming to distinguish reliable predictions from non-reliable ones. Applying our approach towards more fine grained predictions, and in particular considering regression techniques for R_h in order to predict a continuous confidence score, merits further investigation. Finally, integrating our approach within an active learning technique in order to better identify unreliable predictions, seems worth pursuing.

References

[1] Duda R., Hart P., Stork D. Pattern Classification, second edition. (2000) New York: John Wiley & Sons

[2] Freund, Y., Seung, H. S., Shamir, E., Tishby, N. Selective Sampling Using the Query by Committee Algorithm. *Machine Learning*. 28(2), 133 (1997)

[3] Cesa-Bianchi, M., Gentile, C., Zaniboni, L. Worst-Case Analysis of Selective Sampling for Linear Classification. *The Journal of Machine Learning Research*. 7 pp 1205-1230 (2006)

[4] Slonim, N., Crammer, K., Yom-Tov, E. Active online classification via information maximization. *IJCAI*. (2011), in press.

[5] Duda, R. O., Hart P. E. Pattern Classification and Scene Analysis. (1973) New York: John Wiley & Sons

[6] Jain, P., Kapoor, A. Active Learning for Large Multi-class Problems. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*. (2009)

[7] Joshi, A. J., Porikli, F., Papanikolopoulos, N. Multi-class active learning for image classification. *In Proceedings of the International Conference on Computer Vision and Pattern Recognition*. pp. 2372-2379 (2009)

[8] Slonim, N. The Information Bottleneck: Theory and applications. Doctoral dissertation, The Hebrew University of Jerusalem, Jerusalem, Israel (2002)

[9] Cover, T. M., Thomas, J. A. Elements of Information Theory. Wiley (2006)

[10] Hanley, J. A. and McNeil, B. J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*. 143, pp. 2936 (1982)

[11] van Rijsbergen, C. J. Information Retrieval. London: Butterworths (1979)

[12] Newman D.J., Hettich S., Blake C.L., Merz C.J. UCI repository of machine learning databases. (1998).

[13] Zazzi, M. et al. Prediction of response to antiretroviral therapy by human experts and by the EuResist data-driven expert system (the EVE study). *HIV Medicine* 12 pp. 211-218 (2011)

[14] Lin, J. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*. 37(1), 145–151 (1991)

[15] Van Allen T., Greiner R., Hooper P. Bayesian Error-Bars for Belief Net Inference. *In Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. pp. 522-529 (2001).