Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Utility-Based Metaclassification

Michael Pittarelli
SUNY Institute of Technology
Utica, NY 13504-3050, U.S.A.
mike@cs.sunyit.edu

***Abstract*—A technique for accepting or rejecting the output of a Bayesian classifier is presented based on a metalevel Bayesian classifier using features of the probability distribution over the known classes calculated by the lower-level (object-level) classifier. The value of adding the metalevel classifier is determined by user utilities for the possible outcomes of the two-level process.**

***Keywords*—classifier, novelty detection, metaclassification**

I. Introduction

This paper outlines a method for combining an "object-level" (naïve) Bayesian classifier with a metalevel (naïve Bayesian) classifier. The metalevel classifier classifies the lower-level classification as having been either correct or incorrect. While the object-level classifier uses as features properties of the objects it is classifying (income, refractive index, etc.), the metalevel classifier uses features of the object-level probability distribution over the known types (entropy, ratio between highest and second-highest probability, etc.). The metalevel classifier is trained using object-level classifications that are known to have been correct or not, just as the object-level Bayesian classifier is trained using objects of known type.

The metalevel classifier is trained for use in classifying the classifications of a specific object-level classifier for a specific domain, using a specific set of object level features (and a specific method of discretizing the features, a specific set of known types, etc.) When circumstances change at the object level (new types are introduced into the domain, new features are used at the object level, etc.), the metalevel classifier is retrained. An optimal set of features for the metalevel classifier is identified using a wrapper method [1], with maximization of net gain in expected utility as the selection criterion. This requires measurement/elicitation for some agent of the utilities of the possible outcomes of the two-level classification process.

II. A Motivating Example

The "glass identification" data set in the UCI ML Repository [2] contains 214 instances of refractive index and percentage by weight of eight different oxides (magnesium, silicon, etc.) for six types of glass (table glass, headlights, etc.). The data were randomly divided into 175 training cases and 39 test cases, and an exhaustive search for an optimal subset of the nine features was conducted. The maximum accuracy achieved (using various numbers of histogram bins for the features ranging from 16 to 512) was only 67 percent, for a subset consisting of six features. See Figure 1. (The numeric codes used in the UCI data set are retained. There were no samples of glass type 4, vehicle windows that are not float processed, in the original database and therefore none in the test set.)

```
  1    2    3    4    5    6      <-classified as
 ---  ---  ---  ---  ---  ---
 10    3    1    0    0    0      (1): class 1
  2    6    0    2    0    0      (2): class 2
  3    0    5    0    0    0      (3): class 3
  0    0    0    1    0    0      (4): class 5
  0    0    0    0    1    0      (5): class 6
  1    0    1    0    0    3      (6): class 7
```

Figure 1. Confusion matrix for UCI Glass data

Suppose that a system for automated glass type identification is implemented the core of which is a naïve Bayesian classifier based on these six features. Given the performance on the training and test data, it would not be reasonable to feel very confident about the correctness of a classification produced by the system for a glass sample of unknown type.

III. Metaclassification

For the object-level glass type classifier of Section II, an ideal metaclassifier would classify each of the 26 correct classifications as having been correct and would classify each of the 13 incorrect classifications as having been incorrect. For this example, it happens to be possible to achieve this level of performance. To expect this in general from a metaclassifier is as unrealistic as expecting the object level classifier to be 100% accurate. However, just as a feature search at the object level can be tuned to optimize the performance of the classifier in various ways (more severe penalty, in the performance measure, for misclassifying a type 7 glass sample as a type 1 glass sample than for misclassifying a type 2 glass sample as type 5, etc.), the search for optimal sets of metafeatures needn't be based simply on overall accuracy. Given that we cannot expect the metaclassifier to correctly classify all of the object-level classifications, we should bias the search for metafeature sets to take into account the relative desirability of each of the possible metalevel outcomes.

Initially, we will consider only the four generic outcomes: "true positive" (TP), the metaclassifier labels the object-level classification as correct and it was correct; "false positive" (FP), the metaclassifier labels the object-level classification as correct but it was not correct; "true negative" (TN), the metaclassifier labels the object-level classification as incorrect and it was incorrect; and "false negative" (FN), the metaclassifier says the object level was incorrect but it was in fact correct. More specific types of joint object-level/metalevel outcomes can be used, but reliable measurement of the utility associated with these more detailed outcomes may be problematic, if for no other reason than the potential number of them. For example, suppose an object-level classifier labels tumors as benign or malignant. At the metalevel, the classification is labeled as either correct or incorrect. But the metalevel classification can be either correct or incorrect as well. And if we distinguish each of the combinations of error and non-error at both levels, each of the four generic outcomes can be decomposed. Let "R"|"X",Y denote the overall outcome in which the metalevel says that the object level was correct, the object level says that the object is of type X, and the object is actually of type Y. Let 'W' stand for the metalevel indicating that the object level is wrong. For the tumor classifier, let 'M' denote malignant and 'B' denote benign. Then each of the generic outomes is decomposable as follows:

$$TP \equiv \text{"R"}|\text{"B"},B \lor \text{"R"}|\text{"M"},M$$
$$TN \equiv \text{"W"}|\text{"B"},M \lor \text{"W"}|\text{"M"},B$$
$$FP \equiv \text{"R"}|\text{"B"},M \lor \text{"R"}|\text{"M"},B$$
$$FN \equiv \text{"W"}|\text{"B"},B \lor \text{"W"}|\text{"M"},M$$

There is good reason to believe that, for any reasonable agent, the utility of the outcome "R"|"B",M is quite a bit lower (on the usual scale of 0 to 1) than the utility of the outcome "R"|"M",B. In a realistic implementation, if the metalevel says that the lower level is correct in saying the tumor is malignant, but the tumor is actually benign, this will trigger further examination of the tumor (and of other test results, etc.). On the other hand, if the metalevel "certifies" the lower level misclassification of the malignant tumor as benign, there might be no further action taken, the tumor metastasizes, and the patient dies. However, both outcomes are generically lumped together as "false positives". For now, only the generic metalevel outcomes are considered. The basic approach can be adapted to take into account joint metalevel and object-level outcomes at any degree of refinement. The use of more refined joint outcomes (for tumor classification) will be illustrated in Section VII. (The two-level classifier system implemented and currently in use to deal with the actual "motivating example", which is not discussed in this paper, functions very well with only the four generic joint classifier outcomes. Its domain contains dozens of types. There would be thousands of joint outcomes.)

IV. Net Value of Metaclassification

Let 'tp', 'tn', 'fp', 'fn' denote the observed frequency of the outcomes "true positive", etc. We may define three types of accuracy:

object-level accuracy = (tp+fn)/(tp+tf+fp+fn)
metalevel classifier accuracy = (tp+tn)/(tp+tn+fp+fn)
net accuracy = tp/(tp+fp)

Recall that, in this context, FN represents the joint outcome in which the object level is correct but the object level classification is falsely rejected by the metaclassifier, etc. The net accuracy expression is the object-level accuracy calculated after discarding object-level classifications rejected at the metalevel. This quantity can be made arbitrariliy high by biasing the search for metalevel features to reduce the number of false positives, certifying only the "slam-dunk" object level classifications. Hence, it does not seem reasonable to focus on net accuracy as such. The "metalevel classifier accuracy" is the standard concept of accuracy for a binary classifier. It also does not seem to be the quantity we should be attempting to optimize. Selecting metafeatures in a way that increases the likelihood of outcome TP may be accompanied by a smaller increase in FP outcomes, thus increasing metalevel accuracy. But the undesirability of the small increase in FP outcomes may be much greater than the desirability of the larger increase in frequency of TP outcomes, which this ratio does not capture.

Given that errors at the metalevel are inevitable, the search for metaclassifier features should be biased to reflect the preferences of the end user of the two-level classifier system. We measure these preferences as utilities, on a scale of 0 to 1, vs. "cost factors" or "error costs" on an arbitrary scale [3].

Let u(TP), u(TN), u(FP), u(FN) denote the utility associated with each of the four generic joint classification outcomes ("true positive", etc.). These are elicited from someone with both expertise in the domain and an interest in the outcome of the two-level classification procedure. The probabilities of the metaclassification outcomes can be estimated as relative frequencies in searches conducted to evaluate the performance of sets of metalevel classification features.

The net expected utility of the metaclassification process is calculated as the difference between the expected utility over all four possible outcomes of the process and the utility if the metalevel outcomes TP and FN are combined (reflecting the result that the object level was correct, independent of the metalevel classification) and the metalevel outcomes FP and TN are combined (representing the result that the object level was incorrect, independent of the metalevel).

Again let 'tp', 'tn', 'fp', 'fn' denote the observed frequency of the outcomes "true positive", etc., this time in an experiment to evaluate the performance of a set of metalevel features. Let 'total' denote the sum of these frequencies. Then the expected utility of the metaclassification process can be estimated as

$$u(TP) \times tp/total + u(FN) \times fn/total + u(FP) \times fp/total + u(TN) \times tn/total \quad (1)$$

The expected utility of just the object level is estimated as

$$u(TP) \times (tp + fn)/total + u(FP) \times (fp + tn)/total \quad (2)$$

The net value of adding the metalevel is quantity (1) minus quantity (2):

$$u(FN) \times fn/total + u(TN) \times tn/total - u(TP) \times fn/total - u(FP) \times tn/total \qquad (3)$$

The net value (3) can be either positive, negative or zero. For example, if the metalevel fails to overrule the object level in any instance then the gain should evaluate to zero. That it does can be seen by observing that in this circumstance the numerators of each of the fractions in quantity (3) will equal zero.

Some intuitive sense may be made of the net value more generally. Quantity (3) may be rewritten as

$$fn/total \times (u(FN) - u(TP)) + tn/total \times (u(TN) - u(FP)) \qquad (3')$$

The metalevel only has an effect when it rejects the object level clasification. The proportion of the time this occurs is (fn + tn)/total. This can be decomposed as in (3′). The fractional amount of utility lost when the rejection is a false negative is fn/total × u(TP). Similarly, if the agent gives positive utility to false positives, the amount of utility lost via true negatives is tn/total × u(FP).

V. Metafeature Search

For the glass type identification problem of Section II, suppose that the most preferred of the four (generic) types of joint outcome is a true positive. In line with usual practice we assign the value 1 to u(TP). Suppose that the least preferred outcome is a false positive. We therefore assign the value 0 to u(FP). The utilities associated with the non-extreme outcomes can be elicited/assigned in various ways, for example, via a series of questions concerning the choice between an intermediate outcome (FN or TN) for sure and a lottery with probability p of receiving "prize" TP and probability 1 − p of receiving FP. When the probabilities converge to values such that the subject is indifferent between the lottery and the intermediate result, the utility of the intermediate result can be estimated as the probability associated with TP:

$$u(FN) = p \times u(TP) + (1 - p) \times u(FP) \Rightarrow$$
$$u(FN) = p \times 1 + (1 - p) \times 0 \Rightarrow$$
$$u(FN) = p$$

(Similarly for any other combination of most and least preferred outcomes, including situations in which there is a tie for most or least. Also, for our purposes, the utilities need not be measured very precisely. The choice of a set of metafeatures is relatively insensitive to the utility values, which is fortunate. See [4].) Suppose that u(FN) converges (given limits on precision and patience of the subject, etc.) to 0.35 and that u(TN) converges to 0.85, indicating, perhaps, a strong preference for erring on the side of caution when it comes to classifying glass samples based just on the values of these 6 features and the samples on the basis of which the conditional probabilities used by the naïve Bayesian object-level classifier were calculated/estimated. (Note: The system in use, discussed here using for purposes of illustration UCI ML Repository data, assumes a uniform prior distribution over the types. Non-uniform prior probabilities of course could also be estimated from the training samples, subject matter expert knowledge of the domain, etc..)

We now conduct a wrapper-style search for metafeature sets that maximize the net expected utility of adding the metalevel classification stage. Sixteen different metalevel features (of the object-level probability distribution over types) are currently taken seriously: entropy of the entire distribution, maximum probability value, ratio between highest and second-highest probability values, etc. (See list in Section X.) Not all of the features are computable in all instances. (Some require that there are more than two object-level types, etc.) The object-level test instances are transformed into metalevel training instances by assigning to them the metalevel type label "RIGHT" or "WRONG", and substituting the metalevel feature values of the associated object-level probability distribution for the object-level feature values. For example, the object-level test instance

```
33,1.51775,12.85,3.48,1.23,72.97,0.61,8.56,0.09,0.22,1
```

(where 33 is the instance ID number, 1.51775 is the refractive index, etc.) is transformed into the metalevel training instance

```
WRONG,0.309972,16.6667,0.822336,1,0.822336,0.668737,5.35378,68.0719,
0.153599,4.96342,0.0120804,0.988015,4.72544,0.451468,0.30623,26.0158
```

Its first metafeature value, 0.309972, is the entropy of the distribution over the glass types (normalized to the range 0 to 1). Compare this to 0.0450625, the value of entropy for the representative "RIGHT" instance

```
RIGHT,0.0450625,16.6667,0.9866,1,0.9866,0.979184,133.036,166.963,0.007
41605,74.0404,0.0059091,0.999925,649.779,0.0727565,0.447198,355.35
```

Entropy is not a bad indicator of correct classification, for the glass data. However, a search for the best single metafeature reveals that the second-highest probability value is somewhat better, in terms of maximizing net expected utility relative to the fictitious utility values 0.35, etc., above. (See Figure 2, where "Score" is the net gain in expected utility.)

Clearly, there is going to be a strong association between the accuracy of the metaclassifier and its net expected utility, unless the user's utilities are unusual. For example, if we search for metafeatures using u(TP) = 0, u(FN) = 1, u(FP) = 0.5 and u(TN) = 0.5, the result is quite different. Figure 3 shows that there are three single features tied for best in this case (and, of course, none of them is the second-highest probability value), with positive net utility gain of approximately 0.14 utiles, despite an accuracy of only 38% (15/39).

Returning to the more reasonable utilities u(TP) = 1, u(FP) = 0, u(FN) = 0.35, and u(TN) = 0.85, there are 5 2-element sets that are tied for maximum net gain. They also achieve 100% accuracy, as shown in Figure 4. (Note that these spectacular results are based on using all of the object-level test instances as both training and test cases at the metalevel. In the domain motivating this research, there is no shortage of data, and there is as a result no difficulty dividing the metalevel data into disjoint training and test cases and performing crossvalidation, etc.)

```
Best 1-element set(s):
secondhighest
Score = 0.25
Number tied = 1

Classification matrix:

  1    2      <-classified as
---  ---
 24    2      (1): class RIGHT
  0   13      (2): class WRONG
```

Figure 2. Best single metafeature for UCI Glass

```
Best 1-element set(s):

top2ratio
Score = 0.141026
1/(2+3)
Score = 0.141026
(1-2)/(2-3)
Score = 0.141026
Number tied = 3

  1    2      <-classified as
---  ---
  2   24      (1): class RIGHT
  0   13      (2): class WRONG
```

Figure 3. Best single metafeatures implied by ususual preferences

The metaclassifier idea is potentially useful for novelty detection [5] in addition to detecting misclassification of an object whose type is represented in the training set. Recall that the glass data lacks any samples of type 4. Five samples of type 4 glass were simulated by averaging the first test set sample of type 1 with the first samples of each of types 2, 3, 5, 6 and 7, labelling them as type 4 and adding them to the test set. Using the original 175-instance training set (which contains no samples of type 4, bogus or otherwise) and the original 39-instance test set augmented with 5 simulated instances of type 4, the object-level classifier behaves as summarized in Figure 5, using the same set of 6 object-level features that gave rise to the matrix in Figure 1. Note that the number of classification errors rises from 13 to 18. Since there are no instances labeled type 4 in the training set, the five simulated type 4 instances are necessarily classified as other than type 4.

Figure 6 shows the result of using the metafeature set {entropy, 2nd-highest probability} together with histograms generated from the 39 original test cases, which were known to have been either correctly or incorrectly classified at the object level. All five of the bogus type 4 samples are classified as “UNKNOWN”, as are the 13 legitimate samples that are classified incorrectly without the addition of the metalevel.

```
Best 2-element set(s):

entropy
secondhighest
Score = 0.283333

interXile_range
enttopthree
Score = 0.283333

maxprob
enttopthree
Score = 0.283333         1    2     <-classified as
                       ---  ---
secondhighest           26    0     (1): class RIGHT
enttopthree              0   13     (2): class WRONG
Score = 0.283333

enttopthree
entminusmax
Score = 0.283333
Number tied = 5
```

Figure 4. Best 2-element feature sets, reasonable utilities

```
  1    2    3    4    5    6    7      <-classified as
---  ---  ---  ---  ---  ---  ---
 10    3    1    0    0    0    0      (1): class 1
  2    6    0    2    0    0    0      (2): class 2
  3    0    5    0    0    0    0      (3): class 3
  0    0    0    1    0    0    0      (4): class 5
  0    0    0    0    1    0    0      (5): class 6
  1    0    1    0    0    3    0      (6): class 7
  2    2    0    0    1    0    0      (7): class 4
```

Figure 5. Glass samples of type 4 misclassified

```
  1    2    3    4    5    6    7    8  <-classified as
---  ---  ---  ---  ---  ---  ---  ---
 10    0    0    0    0    0    0    4  (1): class 1
  0    6    0    0    0    0    0    4  (2): class 2
  0    0    5    0    0    0    0    3  (3): class 3
  0    0    0    1    0    0    0    0  (4): class 5
  0    0    0    0    1    0    0    0  (5): class 6
  0    0    0    0    0    3    0    2  (6): class 7
  0    0    0    0    0    0    0    5  (7): class 4
  0    0    0    0    0    0    0    0  (8): UNKNOWN
```

Figure 6. Effect of metaclassifier

Again, it is unrealistic to expect such extreme (i.e., perfect) results in general.

VI. A SECOND EXAMPLE

The yeast data set from the UCI ML Repository [2] contains 1484 instances. These were divided randomly into 1215 training cases and 269 test cases. There are 10 classes and 8 predictive attributes. For this particular split (no crossvalidation), the best subset of the 8 features yielded an accuracy of only 56 percent (Figure 7).

```
  1    2    3    4    5    6    7    8    9   10     <-
---  ---  ---  ---  ---  ---  ---  ---  ---  ---
 26    4    5    0    2    1    0    2    2    0    (1): MIT
  3   43   19    0    3    2    1    2    4    0    (2): NUC
  5   21   43    0    2    6    2    2    2    0    (3): CYT
  0    0    0    3    1    0    3    1    0    0    (4): ME1
  0    0    0    1    5    2    1    3    1    0    (5): ME2
  2    0    3    0    0    0    0    0    0    0    (6): VAC
  0    0    1    0    0    0    4    0    0    0    (7): EXC
  1    4    0    0    3    2    0   24    0    0    (8): ME3
  0    0    0    0    0    0    0    0    1    0    (9): POX
  0    0    0    0    0    0    0    0    0    0   (10): ERL
```

Figure 7. Yeast data, best object-level feature subset

Using the same utility values as in Section V, u(TP) = 1, u(FP) = 0, u(FN) = 0.35 and u(TN) = 0.85, the optimal subset contains 8 of the 16 metafeatures, as shown in Figure 8.
All three of the entropy-based metafeatures currently in use appear in this set. All three also appeared among the 2-element sets tied for best performance in the previous example. Unlike the previous example, the metaclassifier does not achieve 100% accuracy.

```
entropy
interXile_range
total_below_Xth_Xile
difftop2
thirdhighest             1    2    <- classified as
1+2+3                  ---  ---
Enttopthree            135   15   (1): class RIGHT
entminusmax              7  112   (2): class WRONG
Score = 0.317658
```

Figure 8. Result of metafeature subset search

Adding the metalevel, the result for this particular training set – test set split is shown in Figure 9. As the matrix in Figure 8 indicates, 112 of the 119 misclassifications at the object level are identified as such by the metaclassifier, but at a cost of 15 false rejections. Focusing on some specific cases, all 16 MIT test cases misidentified at the object level are labelled "UNKNOWN" at the metalevel, at a cost of just one of 26 correctly classified MIT instances being rejected. 33 of the 34 incorrectly classified NUC instances are correctly rejected at the metalevel, at a cost of 6 out of 43 correct classifications incorrectly rejected. On the other hand, the lone POX test case was correctly identified at the object level but falsely rejected at the metalevel.

```
  1    2    3    4    5    6    7    8    9   10   11     <-
 ---  ---  ---  ---  ---  ---  ---  ---  ---  ---  ---
 25    0    0    0    0    0    0    0    0    0   17     (1): MIT
  0   37    1    0    0    0    0    0    0    0   39     (2): NUC
  0    5   37    0    0    0    0    0    0    0   41     (3): CYT
  0    0    0    3    0    0    0    0    0    0    5     (4): ME1
  0    0    0    0    5    0    0    0    0    0    8     (5): ME2
  0    0    1    0    0    0    0    0    0    0    4     (6): VAC
  0    0    0    0    0    0    4    0    0    0    1     (7): EXC
  0    0    0    0    0    0    0   23    0    0   11     (8): ME3
  0    0    0    0    0    0    0    0    0    0    1     (9): POX
  0    0    0    0    0    0    0    0    0    1    0    (10): ERL
  0    0    0    0    0    0    0    0    0    0    0    (11): UNK
```

Figure 9. Effect of metaclassification on yeast data

Unlike the examples discussed in this paper, in the domain for which this technique was developed, object-level features are measured on data samples taken multiple times per second. The probability that successive samples are of the same type is high. For each sample, the top N types receive N, N-1, ..., 1 points. If the metaclassifier rejects the classification of the sample, only "UNKNOWN" receives any points. The classification of the current sample is the type receiving the maximum total points over the most recent K time periods, inclusive. Note that the use of this voting scheme, which injects the metalevel into the object level, can actually result in increased accuracy at the object level when "UNKNOWN" receives votes that would otherwise go to the wrong object.

VII. More Refined Metalevel Outcomes

In Section III, the possibility of refining the ways in which the metalevel classifier could be in error was mentioned, in the context of classifying tumors as benign or malignant. Using the notation introduced in Section III, there are 8 possible joint outcomes: the metalevel classifier classifies the object-level classification as correct, the object level classifier classifies the tumor as benign, and the tumor is benign, denoted "R"|"B",B; the metalevel classifier classifies the object-level classification as incorrect, the object level classification is "malignant", and the tumor is malignant, "W"|"M",M; etc. We need to elicit utilities for each of these 8 outcomes. Suppose that the most preferred joint outcome is "R"|"M",M and the least preferred is "R"|"B",M. Then we assign u("R"|"M",M) the value 1, we assign u("R"|"B",M) the value 0, and we elicit utility values for the remaining 6 outcomes. Suppose that the values converged to are:

u("R"|"B",B) = 0.8   u("R"|"B",M) = 0.0
u("R"|"M",M) = 1.0   u("R"|"M",B) = 0.6
u("W"|"B",M) = 0.8   u("W"|"B",B) = 0.7
u("W"|"M",B) = 0.5   u("W"|"M",M) = 0.1   (4)

Notice that in (4), u("R"|"M",B) ≠ ("W"|"B",B). If both outcomes have exactly the same consequences (same additional testing ordered, etc.), then arguably their utilities should be equal. Enforcing or facilitating consistency and related issues are beyond the scope of this paper, however.

We may estimate the net value of metaclassification, for a particular set of metafeatures (and for a particular set of object level features, etc.), as follows. Let 'rbb' denote the observed frequency (in an experimental setting) of the joint outcome "R"|"B",B, let 'rmm' denote the observed frequency of outcome "R"|"M",M, etc. Let

total = rbb + rmm + …+ wmm.

Then the expected utility of the two-level metaclassification process may be estimated as

u("R"|"B",B) × rbb/total+ …+ u("W"|"M",M) × wmm/total   (5)

The value at the object level alone may be estimated as

u("R"|"B",B) × (rbb + wbb)/total + u("R"|"M",M) × (rmm + wmm)/total +
u("R"|"B",M) ×(rbm + wbm)/total + u("R"|"M",B) ×(rmb + wmb)/total   (6)

The net value of metaclassification is then estimated as the two-level expected utility (5) minus the object-level expected utility (6). The net value can be positive, negative or zero.
As in the more generic approach discussed above, the net value is zero when (but not only when) the metaclassifier fails to reject any of the object-level classifications. In that case,

wbb = wmm = wbm = wmb = 0,

and quantity (6) equals quantity (5).

The Wisconsin Diagnostic Breast Cancer data set [2] contains 569 instances. Each is labelled as "benign" or "malignant". There are 30 real-valued attributes. The data set was divided into 455 training cases and 114 test cases at the object level. An exhaustive search identified a four-element set (concavity standard error, concave points standard error, symmetry standard error, and average of 3 largest symmetry values) as optimal for this particular data split. (A genetic algorithm, greedy algorithm, hybrid algorithms and a parallel exhaustive search algorithm implemented using OpenMP are available for much larger data sets, but are not needed for this example.) The results at the object level for the 114 test cases, using these four features, are summarized in Figure 10.

```
  1    2      <-classified as
---  ---
 40    4      (1): class M
  4   66      (2): class B
```

Figure 10. Breast tumor classification

The probability distributions for the 8 test cases that were misclassified are:

```
MB,0.0220342,0.977966
BM,0.999981,1.91141e-05
BM,0.68534,0.31466
BM,0.986708,0.0132922
BM,0.657708,0.342292
MB,0.493805,0.506195
MB,0.0112428,0.988757
MB,0.112449,0.887551
```

The first number listed is the probability computed for the class "benign". The code "MB" in the first example stands for the classifier having labelled the tumor as malignant when it is in fact benign. Contrast these distributions with the first 8 test cases that were classified correctly:

```
MM,0.016562,0.983438
MM,0.199714,0.800286
MM,0.0084775,0.991522
MM,0.11734,0.88266
MM,0.000901768,0.999098
BB,0.960839,0.0391605
MM,1.5224e-05,0.999985
BB,0.999246,0.000753652
```

On the average, the probability distributions for the misclassified samples are more uniform than those for the correctly classified cases. Almost all of the metafeatures currently in use (see Section X) are measures of uniformity, in some sense (entropy, ratio of highest to second highest probability value, etc.). With just two classes, most of the metafeatures are not computable at all (for example, the entropy of the marginalized distribution over the three highest probability values). Others are computable, but useless for distinguishing correctly classified instances from misclassified ones. For example, the entropy of the distribution resulting from eliminating the highest value and renormalizing will have the value 0 for any probability distribution over two classes. The five metafeatures whose values are computable in the case of two object-level classes and whose values vary with the object-level distribution (entropy, maximum probability, second-highest probability, maximum probability divided by second-highest probability, difference between the highest and second-highest probability) are each derivable from any of the other four (in the case of a 2-component distribution). Therefore, only the maximum probability is used. The result of the metafeature "search" applied to the tumor data (with two classes) is shown in Figure 11, where the string 'rxy' stands for the joint outcome: metalevel identifies object level as correct, object level identifies object as type x, object is really type y. 'wxy' means the metalevel labels object level as incorrect, object level label is x, object is really y. 'm' denotes malignant and 'b' denotes benign.

As shown in Figure 12, use of maximum probability at the metalevel results in rejection of all 4 of the incorrect classifications of benign tumors as malignant. Although two of the 40 correct classifications of malignant tumors are rejected, none of the 66 correct classifications of benign tumors is. Significantly, three of the four incorrect classifications of malignant tumors as benign are flagged as suspect. (Compare the upper left 2×2 submatrix in Figure 12 with the matrix in Figure 10.)

```
Best 1-element set(s):
maxprob
Score = 0.00175439
Number tied = 1
rbb = 66        rbm = 1
rmm = 38        rmb = 0
wbm = 3         wbb = 0
wmb = 4         wmm = 2
```

Figure 11. Metafeature "search" for tumor data

```
  1    2    3      <-classified as
---  ---  ---
 38    1    5      (1): class M
  0   66    4      (2): class B
  0    0    0      (3): class UNKNOWN
```

Figure 12. Metaclassification for tumor data

Note that the use of the maximum probability as a metafeature is consistent with Chow's analysis of error-reject tradeoffs [6]. As an alternative to a naïve Bayes metaclassifier in the case of two classes (e.g., benign vs. malignant), it is possible to search for a probability threshold maximizing net value of metaclassification (as defined here), where a classification is rejected if the probability of the likelier class is below the threshold. Using the object-level distributions for the breast tumor data, the utilities of equations (4), and an increment of 0.0005, probability thresholds in the range 0.5100 to 0.6575 are tied for maximum net utility gain. Further, the gain is greater than the result using Bayesian metaclassification (0.0254 vs. 0.0018). The result is shown in Figure 13.

```
  1    2    3      <-classified as
---  ---  ---
 40    0    4      (1): class M
  1   66    3      (2): class
  0    0    0      (3): class UNKNOWN
```

Figure 13. Reject based on probability threshold

For the yeast data, however, with 10 classes, the optimal probability threshold range of 0.7200 to 0.7210 results in a much lower net gain (0.088 vs. 0.318) than use of the set of metafeatures in Figure 8 (93 false negatives vs. 15, 20 false positives vs. 7). With the refinement of separate thresholds for each class [7], the net gain improves to only 0.172 (vs. 0.318).

VIII. Metametaclassification

In the example of Section VII, the metaclassifier (incorrectly) classified two of the correct object-level classifications of "malignant" as incorrect. In a clinical setting, this would certainly not be the final word. It would be negligent in the extreme in such a case to give a final diagnosis of "benign". It is certain that, instead, additional (medical) tests would be performed, etc.

In other settings, it may not be possible to obtain additional object-level information on the basis of which to revise the object-level classification. In such situations, there may be some value to a second level of metaclassification. The metaclassifier assigns a probability to the events "correct" and "incorrect". The output of the metaclassifier is the event with the higher probability. In an experimental setting, with correctly labelled (object-level) training data, we know whether or not each of the metalevel classifications is correct. We can treat them as object-level classifications for a second level of metaclassification. Since the lower-level (meta)classifier is dealing with only two classes, the metametaclassifier would be limited in the same way that the metaclassifier is in the example of Section VII with respect to choice of metafeatures. (It would be restricted, given the current set of metafeatures, to use of a single feature like maximum probability.) Furthermore, utilities for the four joint outcomes "metametalevel rejects, metalevel correct", "metametalevel rejects, metalevel incorrect", "metametalevel accepts, metalevel correct", "metametalevel accepts, metalevel incorrect" would need to be elicited. These utilities might be more "psychological" in nature, rooted in things like the agent's attitude toward ambiguity, etc.

With a second level of metaclassification, the protocol might be to accept the original classification if the metametaclassifier rejects the metaclassifier's rejection of the original classification, etc.

IX. Conclusion

The concept of a "metaclassifier" for classifying classifications as correct or incorrect was introduced and its use illustrated. It is applicable to Bayesian classifiers and is itself a Bayesian classifier. The features used by the metaclassifier are features of the probability distribution over the lower-level ("object level") classes computed by the object-level classifier. As the examples demonstrate, there is no one set of metafeatures (of the lower level probability distribution) that is optimal for all classification scenarios. An optimal (or satisfactory) set of metafeatures will depend on the lower-level attributes, on the ability of the lower-level attributes to discriminate between the objects, and on the preferences of the agent regarding different types of errors at the metalevel. It is proposed that the strengths of these preferences be measured as utilities and elicited from a domain expert.

The basic method was developed and is intended for use in a setting where large numbers of classifications need to be performed in real time, where there is an abundance of training data at both the object level and the metalevel and where it is possible to determine retrospectively whether a classification was correct or incorrect. There is no "manual handling" of rejected classifications, which is why true negative and false negative metaclassifications are distinguished, vs. a single generic reject outcome with a single cost of rejection.

The term "metaclassification" has been used by many others to refer to the process of combining the output of multiple object-level classifiers (e.g., [8]). These classifiers may be of different types (e.g., a Bayesian classifier combined with a decision tree, etc.) or of the same type (e.g., multiple decision trees combined via bagging or boosting). In this paper we instead propose and illustrate the use of a single "metalevel" Bayesian classifier whose task is to classify the classification of a single "object-level" Bayesian classifier as correct or incorrect.

X. Appendix: List of Metafeatures

A. Entropy.
B. Fraction of values > 1/number of types.
C. Interquantile range, for specified quantile.
D. Total probability below quantile used in feature C.
E. Maximum probability.
F. Maximum probability minus second highest probability.
G. Maximum probability divided by second highest.
H. Maximum probability divided by third highest probability.
I. Second highest probability.
J. Highest divided by second plus third
K. Third highest probability.
L. Sum of three highest probabilities.
M. Highest minus second over third minus fourth
N. Sum of top three divided by sum of next three.
O. Entropy marginalized to top three.
P. Entropy of distribution minus highest value.

References

[1] R. Kohavi and G. John, "Wrappers for feature subset selection," Artificial Intelligence, vol. 97, pp. 273-324, 1997.

[2] A. Asuncion and D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

[3] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk, "Reducing misclassification costs", Proc. 11th Int. Conf. on Machine Learning, pp. 217-225, 1994.

[4] U. Chajewska, D. Koller, and R. Parr, "Making rational decisions with adaptive utility elicitation," Proc. 17th AAAI Conf. on Artificial Intelligence, pp. 363-369, 2000.

[5] M. Markou and S. Singh, "Novelty detection: A review. Part I: Statistical approaches," Signal Processing, vol. 83, pp.2481-2497, 2003.

[6] C. K. Chow, "On optimum recognition error and reject tradeoff," IEEE Transactions on Information Theory, vol. IT-16, pp. 41-46, 1970.

[7] G. Fumera, F. Roli, and G. Giacinto, "Reject option with multiple thresholds," Pattern Recognition, vol. 33, pp. 2099-2101, 2000.

[8] P. Bennett, S. Dumais, and E. Horvitz, "The combination of text classifiers using reliability indicators," Information Retrieval, vol. 8, pp. 67-100, 2005.