A Unified Framework for Evaluation Metrics in Classification Using Decision Trees

Ricardo Vilalta, Mark Brodie,
Daniel Oblinger, and Irina Rish

IBM T.J. Watson Research Center,
30 Saw Mill River Rd., Hawthorne N.Y., 10532 USA
vilalta,brodie,oblinger,rish@us.ibm.com

Abstract. Most evaluation metrics in classification are designed to reward class uniformity in the example subsets induced by a feature (e.g., Information Gain). Other metrics are designed to reward discrimination power in the context of feature selection as a means to combat the feature-interaction problem (e.g., Relief, Contextual Merit). We define a new framework that combines the strengths of both kinds of metrics. Our framework enriches the available information when considering which feature to use to partition the training set. Since most metrics rely on only a small fraction of this information, this framework enlarges the space of possible metrics. Experiments on real-world domains in the context of decision-tree learning show how a simple setting for our framework compares well with standard metrics.

1 Introduction

Evaluation metrics play a critical role in the operation of decision-tree learning algorithms and other symbolic methods for classification, such as rule-based learning and decision lists. An evaluation metric measures the quality in the partitions induced by each of the available features (or functions of features) on a set of training examples; a decision tree is constructed by choosing the highest-quality feature at each tree node. Selecting the right features has strong impact on the final decision tree [8,14,6], contrary to results on earlier studies [9].

Evaluation metrics can be divided in two kinds. The most common kind, *traditional* or *purity-based*, use the proportion of classes on the example subsets induced by each feature; the best result is attained if each example subset is class uniform (i.e., comprises examples of the same class). Examples of traditional or purity-based evaluation metrics are Information Gain and Gain Ratio [11,12], G statistic [9], χ^2 [9,14], Laplace [13], Gini Index [1]. A different class of metrics, *discrimination-based*, quantifies the ability of a feature to separate examples of different class [3,4,7]; most research in this area is found in the context of feature selection as a pre-processing step to classification. Discrimination-based metrics deserve particular attention because of their ability to address the high interaction problem, in which the relevance of a feature can be observed only in combination with other features.

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 503–514, 2001.

This paper introduces a unified framework for evaluation metrics in classification. Our framework is based on a set of parameters and a function of the distance between examples. We show how varying the choice of parameters and distance function allows more emphasis to be placed on either the class uniformity or the discrimination power of the induced example subsets. Our framework, therefore, enables us to combine the strengths of both traditional (or purity-based) and discrimination-based metrics. Furthermore, we show how existing metrics can be defined as instances of this unified framework.

The unification of evaluation metrics in a single framework unveils a space with more metrics than those currently known. This richer characterization of metrics opens the gates to a further understanding of the effects that different splitting functions have during decision-tree learning. Experiments in real-world domains show that the unified framework produces results comparable to using the best of a set of standard evaluation metrics.

The organization of the paper follows. Section 2 provides background information on classification and decision-tree learning. Section 3 reviews the differences between purity and discrimination-based metrics. Section 4 describes our unified framework for evaluation metrics. Section 5 shows experiments on real-world domains using decision trees. Finally, Section 6 presents our conclusions and future work.

2 Preliminaries

A classifier receives as input a set of training examples $T_{\text{train}} : \{(\tilde{\mathbf{X}}_{\mathbf{i}}, c_i)\}$. $\tilde{\mathbf{X}}$ is a feature vector characterized as a point in an n-dimensional feature space, $\tilde{\mathbf{X}} = (X_1, X_2, \cdots, X_n)$. Each feature X_k can take on a different number of values $\{V_m\}$. We refer to the value of feature X_k on vector $\tilde{\mathbf{X}}_{\mathbf{i}}$ as x_k^i, $\tilde{\mathbf{X}}_{\mathbf{i}} = (x_1^i, x_2^i, \cdots, x_n^i)$. $\tilde{\mathbf{X}}_{\mathbf{i}}$ is labeled with class c_i according to an unknown target function or concept C, $C(\tilde{\mathbf{X}}_{\mathbf{i}}) = c_i$ (we assume a deterministic target function, i.e., zero-bayes risk). T_{train} consists of independently and identically distributed (i.i.d.) examples obtained according to a fixed but unknown joint probability distribution Φ in the space of possible feature-vectors $\mathcal{X}$. The goal of the classifier is to produce a hypothesis h that best approximates C, i.e., that minimizes a loss function (e.g., zero-one loss) in the input-output space $\mathcal{X} \times \mathcal{C}$ according to distribution Φ.

In decision-tree learning, each hypothesis takes the form of a tree graph. In its most simple form, the algorithm proceeds top-down; the root of a decision tree is formed by selecting the feature X_k with highest score on T_{train} according to an evaluation metric $\mathcal{M}$. The selected feature splits the training set T_{train} into mutually exclusive subsets $\{T_m\}$, one for each possible feature value. The same methodology is recursively applied to each induced subset, resulting in new subtrees. A node is terminal (i.e., a leaf) if the set of examples T covered by that node are all of the same class, or if the number of examples in T is less than a (user-defined) threshold. A new vector $\tilde{\mathbf{X}}_{\mathbf{i}}$ is classified by starting at the root of the tree and following the branches which match the feature values of $\tilde{\mathbf{X}}_{\mathbf{i}}$ until a leaf is reached.

3 A Review of Evaluation Metrics

3.1 Traditional Metrics

An evaluation metric $\mathcal{M}$ quantifies the quality of the partitions induced by a feature X_k over a set of training examples T. Traditional or purity-based metrics define $\mathcal{M}$ by measuring the amount of class uniformity gained by decomposing T into the set of example subsets $\{T_m\}$ induced by X_k. Let $\tilde{\mathbf{P}}$ be the vector of class probabilities estimated from the data in the complete set T, let $\tilde{\mathbf{P}}_{\mathbf{m}}$ be the corresponding vector of class probabilities estimated from the data in the induced subset T_m, and let I be a measure of the impurity of a class probability vector. $\mathcal{M}$ is typically defined as follows:

$$\mathcal{M}(X_k) = I(\tilde{\mathbf{P}}) - \frac{|T_m|}{|T|}\sum_m I(\tilde{\mathbf{P}}_{\mathbf{m}}) \tag{1}$$

Different variations of $\mathcal{M}$ can be obtained by changing the impurity function I. For example, for Information Gain [11,12], impurity is defined in terms of entropy:

$$\mathrm{I}_{\mathrm{entropy}}(\tilde{\mathbf{P}}) = -\sum_i p_i \mathrm{log}_2 p_i \tag{2}$$

Another example is Gini Index [1], where impurity is defined as follows:

$$\mathrm{I}_{\mathrm{gini}}(\tilde{\mathbf{P}}) = 1 - \sum_i p_i^2 \tag{3}$$

Equation 1 covers most traditional metrics. Other metrics exist in which the class probability vectors are compared with each other, rather than assessed in isolation (e.g., to maximize their degree of orthogonality [2]). Thus, in a general sense, a characterization of traditional metrics can only be obtained by defining $\mathcal{M}$ as an arbitrary function over the class probability vectors[1]:

$$\mathcal{M}(X_k) = F(\tilde{\mathbf{P}}, \{\tilde{\mathbf{P}}_{\mathbf{m}}\}) \tag{4}$$

Traditional metrics exhibit two major limitations. First is a tendency to favor features with many values. Inducing many example subsets increases the probability of finding class-uniform subsets, but at the expense of overfitting. Several solutions have already been proposed for this problem [14,6].

We focus on a second limitation - the inability to detect the relevance of a feature when its contribution to the target concept is hidden by combinations with other features, also known as the feature-interaction problem [3,10]. As an illustration, Figure 1 shows a three-dimensional boolean space where each example can take one of two classes, $\{+, -\}$. The target concept is the Exclusive-OR $X_1 \oplus X_2$, and X_3 is irrelevant (the double arrows will be explained in the next section). Features X_1 and X_2 exhibit high interaction because the class label of an example can be determined only if both features are known.

[1] Strictly speaking $\mathcal{M}$ is also dependent on the size of the example set T, $|T|$. For example both the G and χ^2 statistics depend on $|T|$ in addition to the class probability vectors.

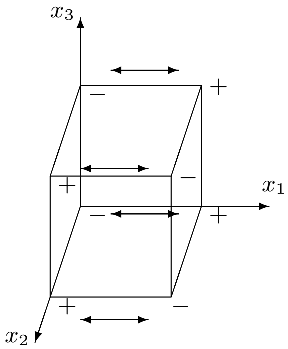

Fig. 1. A 3-dimensional boolean space for target concept $F = X_1 \oplus X_2$, feature X_3 is irrelevant. Features X_1 and X_2 exhibit high interaction: both features are always needed to determine the class label of an example.

In the example above, purity-based metrics fail to give more merit to X_1 and X_2 as compared with X_3, because the class probability vectors are the same for each feature ($\tilde{\mathbf{P}} = \tilde{\mathbf{P}}_{\mathbf{m}} = (0.5, 0.5)$). To attack the feature-interaction problem additional information besides class probabilities is required.

3.2 Discrimination-Based Metrics

A different kind of evaluation metric considers the discrimination power of each feature, i.e., the ability of a feature to separate examples of different class. Let $\tilde{\mathbf{X}}_{\mathbf{i}}$ and $\tilde{\mathbf{X}}_{\mathbf{j}}$ be two examples lying close to each other according to some distance measure D. Feature X_k is awarded some amount of discrimination power if it takes on different values when the class values of $\tilde{\mathbf{X}}_{\mathbf{i}}$ and $\tilde{\mathbf{X}}_{\mathbf{j}}$ differ, i.e., when $x_k^i \neq x_k^j$ and $C(\tilde{\mathbf{X}}_{\mathbf{i}}) \neq C(\tilde{\mathbf{X}}_{\mathbf{j}})$. The more often this condition is true for pairs of nearby examples, the higher the quality of feature X_k.

Let us illustrate how this works for the example in Figure 1. Assume that a feature scores a unit amount whenever the condition $x_k^i \neq x_k^j$ and $C(\tilde{\mathbf{X}}_{\mathbf{i}}) \neq C(\tilde{\mathbf{X}}_{\mathbf{j}})$ is true. As a further simplification, consider only pairs of examples at unit Hamming distance. The double arrows in Figure 1 indicate pairs of neighboring examples where feature X_1 differs in value. Since every time this happens the class of the examples differs, feature X_1 scores a total value of 4. Feature X_2 gets the same score as X_1, whereas X_3 scores a value of 0. Thus discrimination-based metrics take advantage of the distribution of examples in the training set to handle the feature-interaction problem.

Two representative examples of discrimination-based metrics are Contextual Merit [3] and Relief [4,5]. Before describing them, we define the distance between two examples as follows:

$$D(\tilde{\mathbf{X}}_{\mathbf{i}}, \tilde{\mathbf{X}}_{\mathbf{j}}) = \sum_{k=1}^{n} d(x_k^i, x_k^j) \tag{5}$$

For nominal features $d(x_k^i, x_k^j)$ is defined as

$$d(x_k^i, x_k^j) = \begin{cases} 1 & \text{if } x_k^i \neq x_k^j \\ 0 & \text{if } x_k^i = x_k^j \end{cases} \tag{6}$$

For numeric features $d(x_k^i, x_k^j)$ is defined as

$$d(x_k^i, x_k^j) = \frac{|x_k^i - x_k^j|}{\mathrm{TH}(x_k^i, x_k^j)} \tag{7}$$

where TH is a normalization factor, e.g., $\mathrm{MAX}(X_k) - \mathrm{MIN}(X_k)$ (difference between the maximum and minimum values observed for feature X_k in T).

Figure 2 describes the logic behind discrimination-based metrics in a single framework[2]. The algorithm returns a vector of feature scores $\tilde{\mathbf{Q}} = (q_1, q_2, \cdots, q_n)$. For every pair of nearby examples, q_k, the score for feature X_k, is updated as a function of $D(\tilde{\mathbf{X}}_\mathbf{i}, \tilde{\mathbf{X}}_\mathbf{j})$ (equation 5) and $d(x_k^i, x_k^j)$ (equations 6 and 7). Different metrics are obtained by varying the update function (lines 5–6). The Relief algorithm, for example, updates score q_k as follows[3]:

$$q_k = \begin{cases} q_k + d(x_k^i, x_k^j) & \text{if } C(\tilde{\mathbf{X}}_\mathbf{i}) \neq C(\tilde{\mathbf{X}}_\mathbf{j}) \\ q_k - d(x_k^i, x_k^j) & \text{if } C(\tilde{\mathbf{X}}_\mathbf{i}) = C(\tilde{\mathbf{X}}_\mathbf{j}) \end{cases} \tag{8}$$

Thus Relief updates q_k whenever the feature values of two neighbor examples differ; the score increases if their class values differ and decreases if they are the same. Contextual Merit updates q_k when both feature values and class values differ; it uses the update function:

$$q_k = q_k + \frac{d(x_k^i, x_k^j)}{D(\tilde{\mathbf{X}}_\mathbf{i}, \tilde{\mathbf{X}}_\mathbf{j})^2} \text{ if } C(\tilde{\mathbf{X}}_\mathbf{i}) \neq C(\tilde{\mathbf{X}}_\mathbf{j}) \tag{9}$$

In this case the score of a feature decreases quadratically with the distance between two examples.

Discrimination-based metrics have proved effective in the context of feature selection as a pre-processing step to classification. Their design is particularly suitable for domains exhibiting high degree of feature interaction such as protein-folding prediction, board games, parity, etc. These type of metrics, however, ignore the degree of class uniformity of the examples subsets induced by a feature; discrimination power is the only criterion used. We now propose a framework that combines the strengths of both traditional and discrimination-based metrics *during* classification.

[2] We encourage reviewing the references on Relief and Contextual Merit for a detailed understanding of their mechanism. Our description is over-simplified.

[3] The original description of Relief uses the update function $d(x_k^i, x_k^j)/l$ where l is the number of neighbor examples. Since l is a constant we can dispense with it without affecting the final feature ranking.

```
Algorithm 1: Discrimination-Based Metric
Input: Example set $T$
Output: Vector of feature scores $\tilde{\mathbf{Q}}$
COUNT-DISCRIMINATION-POWER($T$)
    Initialize $\tilde{\mathbf{Q}} = (q_1, q_2, \cdots, q_k)$
    foreach example $\tilde{\mathbf{X}}_\mathbf{i} \in T$
        foreach example $\tilde{\mathbf{X}}_\mathbf{j}$ close to $\tilde{\mathbf{X}}_\mathbf{i}$
            foreach feature $X_k$
                Update $q_k$ as a function of
                $D(\tilde{\mathbf{X}}_\mathbf{i}, \tilde{\mathbf{X}}_\mathbf{j})$ and $d(x_k^i, x_k^j)$
    return $\tilde{\mathbf{Q}}$
```

Fig. 2. A general algorithm describing the logic behind discrimination-based metrics.

4 A Unified Framework

To evaluate the quality of feature X_k in our unified framework we extend the strategy of discrimination-based metrics by exploiting additional information between any pair of examples. Recall that feature X_k divides the training set T into a set of subsets $\{T_m\}$, one for each feature value. Figure 3(a) illustrates the possible scenarios in terms of the class agreement between any pair of examples $\tilde{\mathbf{X}}_\mathbf{i}$ and $\tilde{\mathbf{X}}_\mathbf{j}$. The two examples may fall in the same subset (e.g., T_1) and either agree in their class values or not (cases 1 and 2 respectively), or the examples may belong to different subsets (e.g., T_1 and T_2) and either agree in their class values or not (cases 3 and 4 respectively). Although Figure 3(a) shows two classes only, we assume any number of possible classes. Our general approach consists of storing counts for each of these four possible cases separately. Ideally we would like to see high scores for cases 1 and 4, and low scores for cases 2 and 3, since case 1 ($x_k^i = x_k^j$ and $C(\tilde{\mathbf{X}}_\mathbf{i}) = C(\tilde{\mathbf{X}}_\mathbf{j})$) and case 4 ($x_k^i \neq x_k^j$ and $C(\tilde{\mathbf{X}}_\mathbf{i}) \neq C(\tilde{\mathbf{X}}_\mathbf{j})$) ensure the properties of class uniformity and discrimination power respectively, whereas case 2 ($x_k^i = x_k^j$ and $C(\tilde{\mathbf{X}}_\mathbf{i}) \neq C(\tilde{\mathbf{X}}_\mathbf{j})$) and case 3 ($x_k^i \neq x_k^j$ and $C(\tilde{\mathbf{X}}_\mathbf{i}) = C(\tilde{\mathbf{X}}_\mathbf{j})$) work against them.

Our approach works as follows. For each induced example subset T_m, we associate a count matrix R_m. If p is the number of possible class values, each T_m is characterized by a matrix R_m of size $p \times 4$, where row r is a count vector $\tilde{\mathbf{Z}}_\mathbf{r} = (z_{r1}, z_{r2}, z_{r3}, z_{r4})$ which stores the counts for each of the four cases involving examples in class r, as shown in Figure 3(b). In addition we define a weight vector $\tilde{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4)$, $\theta_i \in [0, 1]$, that modulates the contribution of the four counts.

We now explain how to update $\tilde{\mathbf{Z}}_\mathbf{r}$. Given an example $\tilde{\mathbf{X}}_\mathbf{i}$ in class r, for every other example $\tilde{\mathbf{X}}_\mathbf{j}$, exactly one of the four counts z_{ri} is updated, depending on which of the four cases applies to that pair of examples. The appropriate z_{ri} is updated as follows. Given the vector $\tilde{\theta}$ and the function f_α explained below:

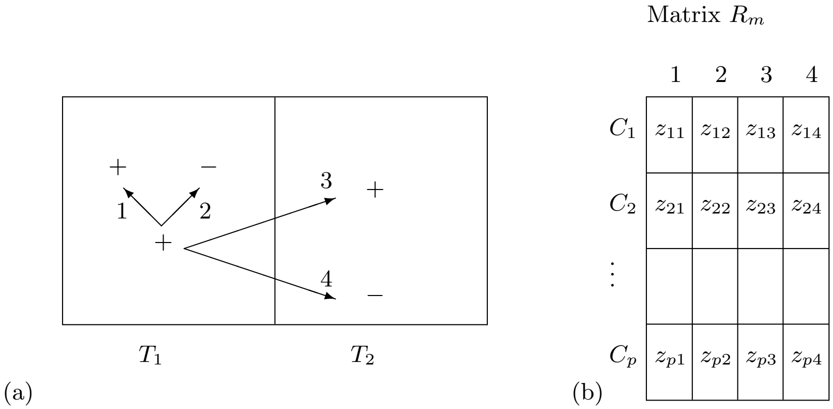

Fig. 3. (a) All four possible scenarios in terms of class agreement for a pair of examples $(\tilde{\mathbf{X}}_\mathbf{i}, \tilde{\mathbf{X}}_\mathbf{j})$. (b) Each count matrix has 4 columns corresponding to the four cases in (a); each row corresponds to a different class.

$$z_{ri} = z_{ri} + \theta_i \cdot f_\alpha(x) \tag{10}$$

where $x = D(\tilde{\mathbf{X}}_\mathbf{i}, \tilde{\mathbf{X}}_\mathbf{j})$ is the distance between the two examples. We assume all features are nominal such that the distance between two feature values may be either zero or one (equation 6). The function f_α decreases with x and may have one of several forms [3]:

$$f_\alpha(x) = \frac{1}{x^\alpha} \quad \text{or} \quad f_\alpha(x) = \frac{1}{2^{\alpha \cdot x}} \tag{11}$$

Large values for α narrow our attention to only the closest neighboring examples. Small values for α extend our attention to examples lying far apart. In the extreme case where $\alpha = 0$ all examples are considered equally, irrespective of distance. Thus α enables us to vary the relative importance of the distance between any two examples.

The vector $\tilde{\theta}$ modulates the degree of contribution of each of the four cases in Figure 3. In particular, setting θ_i to zero nullifies the contribution of the ith case. We will show how varying the values of $\tilde{\theta}$ allows us to put more weight on either class uniformity or discrimination power.

Figure 4 describes the computation of the set of matrices $\{R_m\}$. In essence, every example is compared against all other examples in T, while the counts for each matrix R_m are updated. For simplicity the algorithm is described for a single feature X_k, but the double loop in lines $2-3$ can be done over all features. We select a matrix R_m according to the value of feature X_k in $\tilde{\mathbf{X}}_\mathbf{i}$. The row index corresponds to the class value of $\tilde{\mathbf{X}}_\mathbf{i}$, $C(\tilde{\mathbf{X}}_\mathbf{i})$. The column index corresponds to the case to which $\tilde{\mathbf{X}}_\mathbf{i}$ and $\tilde{\mathbf{X}}_\mathbf{j}$ belong (Figure 3(a)). Once the matrix entry is located, the corresponding z_i is updated according to equation 10.

Lines $2-3$ in Figure 4 cycle through all examples in T. There is no need to limit the second loop to the closest examples (as in Algorithm 1) because

```
Algorithm 2: Unified Framework
Input: Example set $T$, Feature $X_k$
Output: Set of matrices $\{R_m\}$
UPDATE-MATRICES($T$,$X_k$)
    Initialize all matrices in $\{R_m\}$
    foreach example $\tilde{\mathbf{X}}_\mathbf{i} \in T$
        foreach example $\tilde{\mathbf{X}}_\mathbf{j} \in T$
            Let $C(\tilde{\mathbf{X}}_\mathbf{i}) = c_r$ and $x_k^i = V_m$
            Update $R_m[r, \cdot]$ using the
            corresponding $z_{r\cdot}$ in eq. 10
return $\{R_m\}$
```

Fig. 4. Logic describing how to compute the set of count matrices for feature vector X_k on example set T.

the update function depends on distance and is regulated by parameter α (we explain later why we allow comparison of pairs of identical examples).

The training set T also gives rise to a matrix R, as a function of the set $\{R_m\}$, but because examples in T cannot be compared to different example sets all columns in R corresponding to cases 3 and 4 must equal zero. Our evaluation metric evaluates the quality of a feature X_k as a function of the matrix R for the training set T and the matrix R_m for each of the induced subsets $\{T_m\}$ (computed as shown in Figure 4):

$$\mathcal{M}(X_k) = F(R, \{R_m\}) \tag{12}$$

Finally, our unified framework for evaluation metrics Π is a 4-tuple containing all the parameters necessary to define a metric of the form defined in equation 12:

$$\Pi = (F, \tilde{\theta}, \alpha, f_\alpha) \tag{13}$$

Complexity Analysis. The algorithm in Figure 4 runs in time $O(n\ t^2)$, where n is the number of features and $t = |T_{\mathrm{train}}|$ is the size of the training set. If the algorithm is used at every node of a decision tree, the complexity is $O(l\ n\ t^2)$ where l is the number of nodes in the tree. This contrasts with traditional metrics where the complexity is usually $O(l\ n\ t)$. One natural extension is to pre-compute, for each example, an ordered list of examples based on distance, with a complexity of $O(t^2\ log\ t)$. Then, during decision-tree learning, one can find the (fixed) k closest examples to each example within the induced partition in time $O(l\ n\ t\sqrt{t})$. The solution[4] is feasible only for small k. Another solution for large datasets ($t \sim 10^3 - 10^4$), is to use only a sample S of the training set to compute the count matrices. Let $s = |S|$, the complexity is reduced to $O(l\ n\ s^2)$.

[4] The $\sqrt{t}$ term is derived from a worst case analysis (using the best of two different methods) for obtaining the k closest examples.

4.1 Instances of the Unified Framework

We now show how our unified framework for evaluation metrics covers traditional, or purity-based metrics (Section 3.1), and also discrimination-based metrics (Section 3.2).

Proposition 1. For a specific setting on the parameters of framework Π, it is possible to derive all traditional metrics.

Proof. Function F in equation 4 is left undefined; F defines how to measure the quality of a feature based on class proportions. We simply show that for a specific setting of Π we can derive all class proportions. Consider the result of running Algorithm 2 with $\tilde{\theta} = (1, 0, 0, 0)$. Since we care about class uniformity only (Figure 3, Case 1), we consider only pairs of examples with the same class value and the same feature value. Assume $f_\alpha(x = 0) = 1$ and $f_\alpha(x \neq 0) = 0$ (x is the distance $D(\tilde{\mathbf{X}}_\mathbf{i}, \tilde{\mathbf{X}}_\mathbf{j})$ between the two examples). Since $f_\alpha(x) = 1$ only when the distance between examples is zero, our comparisons are limited to pairs of identical examples. Therefore, the counts on each matrix R_m are zero in columns $2 - 4$, and column 1 reflects the number of examples of each class when the feature value is fixed. These counts are sufficient to compute F: class counts can be easily converted into class proportions by dividing over the sum of all entries in column 1, i.e., by dividing over $\sum_i R_m[i, 1]$. This completes the proof.

Proposition 2. Both Relief and Contextual Merit can be defined as instances of framework Π.

Proof. We begin with Contextual Merit. Consider the result of running Algorithm 2 with $\tilde{\theta} = (0, 0, 0, 1)$, $\alpha = 2$, and $f_\alpha = \frac{1}{x^\alpha} = \frac{1}{x^2}$. We now care about discrimination power exclusively (Figure 3, Case 4), and compare examples with different class value and different feature value. The counts on each matrix R_m are zero on columns $1 - 3$; the sum of the values along column 4 over all $\{R_m\}$, $\sum_m(\sum_i(R_m[i, 1])$, is exactly the output produced by Contextual Merit when each example in T is compared against all other examples (i.e., when lines 2-3 in Algorithm 1 run over all examples).

We now look into Relief. Consider the result of running Algorithm 2 with $\tilde{\theta} = (0, 0, 1, 1)$ and $f_\alpha(x) = 1$ if $x < \alpha$ and 0 otherwise; α takes the role of defining a threshold that allows comparison of only the α-nearest neighbors. Since $\tilde{\theta} = (0, 0, 1, 1)$, we favor discrimination power but penalize working against it. We compare examples with different feature value irrespective of class value. The counts on each matrix R_m are zero in columns $1 - 2$; the sum of the values along column 4 over all $\{R_m\}$ minus the respective sum along column 3, $\sum_m(\sum_i(R_m[i, 4] - R_m[i, 3])$, is the output produced by Relief for the appropriate value of α. This completes the proof.

The unified framework Π adds versatility to our new family of metrics by enabling us to modulate how much emphasis should be placed on class uniformity (or lack thereof) and discrimination power (or lack thereof). We now measure empirically the effects of a simple settings for Π in decision-tree learning.

5 Experiments

Our experiments measure the performance accuracy for a simple setting of Π and compare it to the performance of a set of standard evaluation metrics. We adopt a simple model for F by adding the values, over all matrices in $\{R_m\}$, in columns 1 and 4, and subtracting the values in columns 2 and 3. We do this for each feature value and then take the weighted average according to the number of examples in each example subset:

$$F = \frac{|T_m|}{|T|} \sum_m G(R_m) \tag{14}$$

$G(R_m)$ is defined as follows:

$$G(R_m) = \sum_{i=1}^{p} (R_m[i,1] + R_m[i,4] - R_m[i,2] - R_m[i,3] \tag{15}$$

where p is the number of classes. The definition for $G(R_m)$ corresponds to $\tilde{\theta} = (1,1,1,1)$, which can be regarded as a compromise between class purity and discrimination power. For the update function, we consider $f_\alpha = \frac{1}{2^{\alpha \cdot x}}$ and $\alpha = 0.1$.

5.1 Methodology

We test our model using a decision-tree learning algorithm. An initial discretization step on all numeric features divides each feature domain into ten equally-sized intervals. We stop growing a tree if the number of examples on a node is less than $\beta = 3$ or if all examples are class uniform. The final tree is pruned using a pessimistic-pruning method [12].

All experiments use real-world domains extracted from the UCI repository. Predictive accuracy is estimated by averaging over five repetitions using 10-fold cross validation. Tests of significance use a two-sided Student-t distribution at the $p = 0.05$ level.

5.2 Results on Real-World Domains

Our experimental results are depicted in Table 1 (numbers enclosed in parentheses represent standard deviations). For each row, the best score is highlighted in bold style; an asterisk on the left implies a significant difference. Among the set of standard metrics, the subset corresponding to purity-based metrics (Gini and Information Gain; columns 2 and 3 respectively) tend to perform, on average (last row), slightly better than the subset corresponding to discrimination-based metrics (Contextual Merit and Relief; columns 4 and 5 respectively). This may simply indicate that most domains in the UCI repository exhibit a low degree of feature interaction. Our framework scores best on eight of the eighteen domains, outperforming the second-best metric significantly on 4 of these. Performance is, on average, better than other metrics (not significantly).

Table 1. A comparison of decision-tree accuracy using different evaluation metrics. Numbers enclosed in parentheses represent standard deviations.

Domain	Gini	Info Gain	C. Merit	Relief	Instance of Π
bupa	57.86 (0.31)	58.54 (2.78)	57.40 (0.65)	58.50 (1.80)	***62.32** (1.29)
cancer	93.64 (0.30)	**95.02** (0.15)	95.00 (0.25)	93.10 (0.27)	92.56 (0.38)
credit	72.80 (1.35)	67.80 (2.57)	66.56 (1.03)	70.82 (1.89)	***74.74** (0.67)
heart	***78.62** (1.48)	76.06 (1.52)	66.14 (1.18)	71.74 (2.48)	75.68 (1.26)
hepatitis	81.82 (1.41)	81.90 (1.36)	82.66 (0.45)	83.04 (2.23)	**83.26** (0.61)
ionosphere	90.32 (0.75)	**90.46** (0.48)	78.90 (0.50)	81.64 (1.05)	86.62 (0.56)
chess-end	98.14 (0.52)	**98.42** (0.50)	97.38 (0.47)	88.74 (1.48)	94.96 (0.72)
lymphography2	79.28 (1.69)	**81.88** (1.82)	74.02 (2.13)	74.44 (3.00)	81.50 (1.89)
lymphography3	77.64 (2.03)	**78.90** (2.13)	73.30 (2.10)	77.54 (3.02)	77.02 (1.31)
mushroom	98.36 (0.63)	98.84 (0.27)	98.64 (0.16)	99.50 (0.36)	**99.52** (0.48)
thyroid-hyper	89.22 (0.30)	94.34 (0.33)	93.24 (0.77)	92.96 (0.88)	**94.44** (0.29)
thyroid-hypo	86.08 (0.04)	92.36 (0.48)	86.44 (0.69)	89.68 (1.48)	***93.70** (0.35)
diabetes	64.68 (0.30)	73.42 (0.25)	***74.28** (0.48)	66.38 (0.66)	68.84 (0.73)
promoters	74.36 (2.15)	74.94 (4.64)	73.20 (3.80)	**76.36** (2.68)	76.08 (3.29)
star-cluster	80.64 (0.46)	80.64 (0.46)	80.64 (0.46)	78.32 (2.75)	**81.36** (0.82)
tic-tac-toe	82.96 (0.69)	85.62 (1.26)	83.06 (1.07)	79.42 (1.47)	***87.94** (0.23)
voting	95.20 (0.13)	94.78 (0.31)	***95.44** (0.15)	93.38 (0.88)	95.16 (0.08)
zoo	94.76 (0.78)	**95.16** (0.08)	95.16 (0.08)	94.76 (0.78)	94.76 (0.78)
Average	**83.13**	**84.39**	**81.75**	**81.68**	**84.47**

Other empirical tests (not shown here for space considerations) illustrate the effect of varying $\tilde{\theta}$ (Section 4). Results show significant differences in accuracy, but no single setting for $\tilde{\theta}$ outperforms other settings consistently. In addition, our framework is sensitive to function $f_\alpha(x)$. Varying $f_\alpha(x)$ while keeping other parameters fixed results in significant differences in accuracy but to a lesser degree than varying $\tilde{\theta}$.

6 Conclusions and Future Work

We have defined a novel framework for evaluation metrics in classification. Our framework enriches the information derived when a feature is used to partition the training set T by capturing all possible scenarios in terms of class agreement (or disagreement) between pairs of examples in T. Most metrics utilize only a small fraction of the information contained in Π; our framework, therefore, provides a broader view of the space of possible metrics. A limitation of our approach is the complexity of the algorithm that generates count matrices (Figure 4); proposed solutions include a pre-processing step, or using a sample of the training set (for large datasets). Experiments using real-world domains show our unified framework compares well in performance with the best of a set of standard metrics in the context of decision-tree learning. Our results imply that the best metric for any given problem may not be one of the few instances previously studied. Placing them in a parameterized framework is important–it delineates a much larger and richer space of alternatives.

A line of future research is to explore whether there exists generally useful instances in the new space of metrics that could be applied to a wider class of learning tasks. Another line of research consists of trying to match domain characteristics with the appropriate parameter settings in Π (equation 13). The flexibility inherent in our unified framework in finding a balance among several criteria suggests guiding the parameter settings according to the characteristics (i.e., meta-features) of the domain under analysis. For example, meta-features could be functions of the counts in the matrix R over the set T, where T corresponds to the whole training set T_{train} (Section 4). Those counts provide information about the domain itself and relate directly to Π.

Acknowledgments. We are grateful to Vittorio Castelli for his valuable help and suggestions. This work was supported by IBM T.J. Watson Research Center.

References

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and Regression Trees. Wadsworth, Belmont, CA (1984).
2. Fayyad, U., Irani, K. The Attribute Selection Problem in Decision Tree Generation. Conference of the American Association for Artificial Intelligence (1992), 104–110.
3. Hong, S. J.: Use of Contextual Information for Feature Ranking and Discretization. IEEE Transactions of Knowledge and Data Engineering (1997).
4. Kira, K., Rendell, L.: A practical approach to feature selection. Ninth International Workshop on Machine Learning. Morgan Kaufmann Publishers (1992), 249-256.
5. Kononenko, I.: Estimating attributes: analysis and extensions of RELIEF. European Conference on Machine Learning. Springer Verlag (1994), 171-182.
6. Kononenko, I: On Biases in Estimating Multi-Valued Attributes. International Joint Conference on Artificial Intelligence (1995), 1034-1040.
7. Kononenko, I, Hong, S.J.: Attribute Selection for Modeling. Future Generation Computer Systems (1997).
8. Liu, W.Z., White, A.P.: The Importance of Attribute Selection Measures in Decision Tree Induction. Machine Learning (1994), 15, 25-41.
9. Mingers, J.: An Empirical Comparison of Selection Measures for Decision-Tree Induction. Machine Learning (1989), 3, 319–342.
10. Pérez, E., Rendell, L. A.: Using Multidimensional Projection to Find Relations, Twelfth International Conference on Machine Learning (1995), 447-455.
11. Quinlan, J. R.: Induction of Decision Trees. Machine Learning (1986), 1, 81-106.
12. Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1994).
13. Quinlan, J. R.: OverSearching and Layered Search in Empirical Learning. International Joint Conference on Artificial Intelligence. Morgan Kaufmann (1995), 1019-1024.
14. White, A.P., Liu, W.Z.: Bias in Information-Based Measures in Decision Tree Induction. Machine Learning (1994), 15, 321-329.