Generalized Unified Decomposition of Ensemble Loss

Remco R. Bouckaert[1,2], Michael Goebel[3] and Pat Riddle[3]

1. Xtal Mountain Information Technology, Auckland, rrb@xm.co.nz
2. Computer Science Department, University of Waikato, remco@cs.waikato.ac.nz
3. Department of Computer Science, University of Auckland, New Zealand
{mgoebel,pat}@cs.auckland.ac.nz

Abstract. Goebel et al. [4] presented a unified decomposition of ensemble loss for explaining ensemble performance. They considered democratic voting schemes with uniform weights, where the various base classifiers each can vote for a single class once only. In this article, we generalize their decomposition to cover weighted, probabilistic voting schemes and non-uniform (progressive) voting schemes. Empirical results suggest that democratic voting schemes can be outperformed by probabilistic and progressive voting schemes. This makes the generalization worth exploring and we show how to use the generalization to analyze ensemble loss.

This article is inspired by Goebel et al. [4], hereafter referred to as UDEL to reflect the title **U**nified **D**ecomposition of **E**nsemble **L**oss. They decomposed the loss of a classifier ensemble into the mean loss of the individual ensemble members and a loss term D which is a measure of the diversity of the ensemble members. However, they considered only ensembles where the various base classifiers each can vote for a single class once only. Here a GUDEL, a generalized UDEL is presented. This generalized decomposition additionally applies to ensembles with weighted votes, as well as to ensembles where the member classifiers output a probability distribution instead of a single vote. We also investigate several other voting schemes which are covered neither by UDEL or by GUDEL. Experiments show that simple democratic voting schemes can sometimes be outperformed by other voting schemes [1, 5].

Preliminaries: Given input space $\mathbf{X} = \mathbf{X}_1 \times ... \times \mathbf{X}_v$, a set of class labels $Y = \{Y_1, ..., Y_n\}$, and an unknown but stationary probability distribution P over $\mathbf{X} \times Y$, a learner is usually given a finite sample $D = \{\langle \mathbf{x}_1, y_1 \rangle, ..., \langle \mathbf{x}_m, y_m \rangle\}$ drawn from $\mathbf{X} \times Y$ according to P and is required to produce a classifier c. Given a (previously unseen) test example $\mathbf{x} = \langle x_1, ..., x_v \rangle$, this classifier produces a prediction $\widehat{y}_c(\mathbf{x})$. The performance of models (and therefore implicitly the learners that produced those models) is measured using loss functions: A loss function $l : Y \times Y \rightarrow \Re$ measures the cost of making the prediction $\widehat{y}$ when the true value is y. For the case where Y is a set of class labels $Y = \{Y_1, ..., Y_n\}$, the most commonly used loss function is the zero-one-loss ($l_{01}(\widehat{y}, y) := 0$ iff $\widehat{y} = y$; $l_{01}(\widehat{y}, y) := 1$ otherwise). The goal of a learner should be to produce a model with

the smallest possible expected loss; i.e., a model which minimizes the average loss over examples drawn independently from $\mathbf{X} \times Y$ according to P. Some classifiers, rather than producing a prediction $\widehat{y}_c(\mathbf{x})$ directly, output instead a probability distribution $\widehat{P}_c(Y|\mathbf{x})$ over the set of class labels Y, such that, for any $y \in Y$, $\widehat{P}_c(y|\mathbf{x})$ reflects the degree of the classifier's internal belief that the test example $\mathbf{x}$ belongs to class y. In those cases, the ensemble prediction $\widehat{y}(\mathbf{x})$ is taken to be $\widehat{y}(\mathbf{x}) = \arg\min_{y' \in Y} \sum_{c \in C} w_c \int_{y'' \in Y} \widehat{P}_c(y''|\mathbf{x}) l(y'', y') dy''$.where C is an ensemble, and w_c the weight of the classifier c in the ensemble. This is the minimum average distance in terms of the loss function of the member predictions weighted by the members belief and their weigths.

Voting Schemes: A typical ensemble algorithm under 0-1 loss takes the predictions of the base classifiers giving the prediction $\widehat{y}(\mathbf{x}) := \arg\max_{y \in Y} \sum_{c \in C} w_c I(y = \widehat{y}_c(\mathbf{x}))$where $I(\cdot)$ is the indicator function ($I(y = \widehat{y}_c) = 1$ iff $\widehat{y}_c$ equals y, and 0 otherwise). When two or more classes get the same vote, one is chosen at random. This voting scheme is called *democratic voting* also known as *majority vote*. The base classifiers predictions are possibly weighted with a weight w_c. Throughout this article, we assume the weights are normalized to 1, i.e., $\sum_{c \in C} w_c = 1$.

A lot of commonly used base classifiers calculate a probability distribution $\widehat{P}_c(y|\mathbf{x})$ over y and pick the class with the highest probability as their prediction, for instance, decision trees, naive Bayes, and Bayesian network variants. So, instead of taking the class with the highest probability as a vote, one can take the class receiving the maximum average probabililty, where the average is taken over the weighted member classifiers. Formally, this gives $\widehat{y}(\mathbf{x}) := \arg\max_{y \in Y} \sum_{c \in C} w_c \widehat{P}_c(y|\mathbf{x})$where $\widehat{P}_c(y|\mathbf{x})$ is the probability that classifier c assigns to class y given observation $\mathbf{x}$. This voting scheme will be called *probabilistic voting*. Alternatively, it can be argued that the model with the highest confidence in its ability to classify should be making the decision. This *aristocratic voting* scheme returns the class $\widehat{y}(\mathbf{x}) := \arg\max_{y \in Y} (\max_{c \in C} w_c \widehat{P}_c(y|\mathbf{x}))$

Balancing aristocratic and democratic arguments, the vote can be weighed according to a convex function of $\widehat{P}_c$, for example quadratic or exponential. The *progressive quadratic voting* scheme results in the following classification: $\widehat{y}(\mathbf{x}) := \arg\max_{y \in Y} \sum_{c \in C} w_c (\widehat{P}_c(y|\mathbf{x}))^2$and *progressive exponential voting* scheme: $\widehat{y}(\mathbf{x}) := \arg\max_{y \in Y} \sum_{c \in C} w_c e^{\widehat{P}_c(y|\mathbf{x})}$The standard bagging algorithm [3] uses the democratic voting scheme. It can be easily adapted to apply each of the above voting schemes by plugging in the appropriate voting scheme.

Decomposition: First, we need to define some terms. The *ensemble prediction* of an ensemble C for input $\mathbf{x}$ is denoted as $\widehat{y}(\mathbf{x})$ or $\widehat{y}$ if the input $\mathbf{x}$ is clear from the context. The way the ensemble prediction is calculated depends on the voting scheme applied.

Note that UDEL defines the ensemble prediction in terms of a loss function, while our definition does not take this into account. This specializes to the definition used by [4], which was $\widehat{y}(x) := \arg\min_{y \in Y} E_{c \in C}[l(y, \widehat{y}_c(x))]$, with $w_c := 1/|C|$ and $\widehat{P}_c(y'|\mathbf{x}) := 1$ iff $y' = \widehat{y}_c(\mathbf{x})$; $\widehat{P}_c(y'|\mathbf{x}) := 0$ otherwise for all $c \in C$ and $y' \in Y$.

Definition 1. *The* loss *of ensemble C on instance $\langle \mathbf{x}, y \rangle$ under loss function l is given by $L(\langle \mathbf{x}, y \rangle) := l(\widehat{y}(\mathbf{x}), y)$. The* expected loss *of ensemble C on the domain $\langle \mathbf{X}, Y, P \rangle$ is given by $L := E_P[L\,\langle \mathbf{x}, y \rangle] = \int_{X \times Y} L(\langle \mathbf{x}, y \rangle) p(\langle \mathbf{x}, y \rangle)\, d\,\langle \mathbf{x}, y \rangle$.*

Typically, we are interested in learning ensembles that have a low expected loss, where the expectation is taken over the instance distribution P.

Definition 2. *The* mean member loss *of ensemble C on instance $\langle \mathbf{x}, y \rangle$ under loss function l is given by $\overline{L}(\langle \mathbf{x}, y \rangle) := \sum_{c \in C} w_c E_{\widehat{P}_c}[l(\widehat{y}_c(\mathbf{x}), y)]$ which equals $\sum_{c \in C} \int_{y' \in Y} w_c l(y', y) \widehat{P}_c(y' | \mathbf{x}) dy'$.*

The expected mean member loss *of ensemble C on the domain $\langle \mathbf{X}, Y, P \rangle$ is given by $\overline{L} := E_P[\overline{L}\,\langle \mathbf{x}, y \rangle] = \int_{X \times Y} \overline{L}(\langle \mathbf{x}, y \rangle) p(\langle \mathbf{x}, y \rangle) d\,\langle \mathbf{x}, y \rangle$.*

The mean member loss indicates how much the predictions of the individual base classifiers in the ensemble C differ from the true value of the class y. This definition differs from UDEL in that the classifier weights w_c are taken into account when taking the expectation over the set of classifiers C. This specializes to the definition in UDEL when $w_c = 1/|C|$ for all classifiers $c \in C$. Also, the expectation is taken over both C and Y while in UDEL only the expectation over C is taken. This will take into account their class probabilities instead of just their final predictions in the case where the base classifier is a distribution classifier. If the base classifier is not a distribution classifier, the definition still applies with $\widehat{P}_c(y' | \mathbf{x}) = I(y' = \widehat{y}(\mathbf{x}))$, where $I(\cdot)$ is the indicator function, i.e, $I(y' = \widehat{y}(\mathbf{x})) := 1$ iff y' equals $\widehat{y}(\mathbf{x})$, and 0 otherwise. In this case, the definition of mean member loss coincides with the one from UDEL, which was $\overline{L}(\langle x, y \rangle) := E_{c \in C}[l(\widehat{y}_c(x), y)]$.

Definition 3. *The* diversity *of ensemble C on input $\mathbf{x}$ under loss function l is $\overline{D}(\mathbf{x}) := \sum_{c \in C} w_c E_{\widehat{P}_c}[l(\widehat{y}_c(\mathbf{x}), \widehat{y}(\mathbf{x}))] = \sum_{c \in C} \int_{y' \in Y} w_c l(y', \widehat{y}(\mathbf{x})) \widehat{P}_c(y' | \mathbf{x}) dy'$.*

The expected diversity *of ensemble C on the domain $\langle \mathbf{X}, Y, P \rangle$ is given by $\overline{D} := E_P[\overline{D}(\mathbf{x})] = \int_{X \times Y} \overline{D}(\mathbf{x}) p(\langle \mathbf{x}, y \rangle) d\,\langle \mathbf{x}, y \rangle$.*

The diversity indicates how much the predictions of the individual base classifiers in the ensemble C differ from the ensemble prediction $\widehat{y}(\mathbf{x})$. Again, this definition differs from UDEL in that here the classifier weights w_c are taken into account, and in that the expectation is taken over both C and Y while in UDEL only the expectation over C is taken, such as $\overline{D}(x) := E_{c \in C}[l(\widehat{y}_c(x), \widehat{y}(x))]$.

Figure 1 shows the relations between the various terms just defined. It shows that the loss L is a function of the real class y and the ensemble prediction $\widehat{y}$, the expected mean member loss $\overline{L}$ is a function of y and the various base classifiers predictions $\widehat{y}_c$, and the expected diversity $\overline{D}$ is a function of the various base classifiers predictions $\widehat{y}_c$ and the ensemble prediction $\widehat{y}$.

Following UDEL, a decomposition of L is proposed as a function of $\overline{L}$ and $\overline{D}$, that is,

$$L\,(\langle \mathbf{x}, y \rangle) = f(\overline{L}(\langle \mathbf{x}, y \rangle), \overline{D}(\mathbf{x})) \tag{1}$$

for ensembles of weighted, probabilistic classifiers. This covers ensembles voting democratically or probabilistically with or without individual classifier weights.

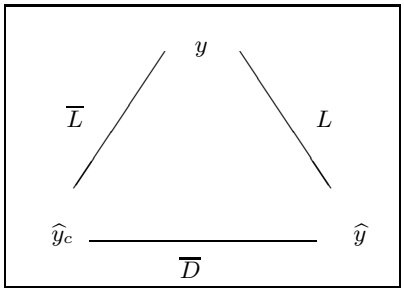

Fig. 1. Intuitive relations between L, $\overline{L}$, $\overline{D}$ and y, $\widehat{y}$ and $\widehat{y}_c$.

Though it does not cover the aristocratic or progressive voting schemes the theory can be easily extended by normalizing the votes in the schemes so that the votes add to unity for each particular instance. The extensions to the theory which cover these cases are discussed in the full paper. In the following section, details for 0-1 loss will be given.

Instantiating the decomposition for 0-1 loss: We first instantiate Equation 1 for the case where the class is binary, that is $Y = \{0, 1\}$, and the loss function l is the 0-1 loss, i.e., $l(y_1, y_2) = I(y_1 \neq y_2)$.

Lemma 1. *For 0-1 loss in two-class problems, the ensemble loss* $L(\langle \mathbf{x}, y \rangle)$ *can be written as* $L(\langle \mathbf{x}, y \rangle) = f(\overline{L}(\langle \mathbf{x}, y \rangle), \overline{D}(\mathbf{x})) = \overline{L}(\langle \mathbf{x}, y \rangle) + k(x, y)\overline{D}(\mathbf{x})$ *where* $k(x, y) = -1$ *iff* $\widehat{y}(\mathbf{x}) = y$ *and* $k(x, y) = 1$ *iff* $\widehat{y}(\mathbf{x}) \neq y$.

Refer to [2] for proofs of this and following lemmas.

For a given ensemble C, let $T := \{\langle \mathbf{x}, y \rangle \,|\widehat{y}(\mathbf{x}) = y\}$ be the set of instances which the ensemble classifies correctly, and let $F := \{\langle \mathbf{x}, y \rangle \,|\widehat{y}(\mathbf{x}) \neq y\}$ be the set of instances which the ensemble classifies incorrectly. Then, we can define $\overline{D}_T$ as the diversity over T and $\overline{D}_F$ as the diversity over F. More formally, $\overline{D}_T := \int_{\langle \mathbf{x}, y \rangle \in T} \overline{D}(\mathbf{x}) p(\langle \mathbf{x}, y \rangle \mid \langle \mathbf{x}, y \rangle \in T) d\langle \mathbf{x}, y \rangle$ and $\overline{D}_F := \int_{\langle \mathbf{x}, y \rangle \in F} \overline{D}(\mathbf{x}) p(\langle \mathbf{x}, y \rangle \mid \langle \mathbf{x}, y \rangle \in F) d\langle \mathbf{x}, y \rangle$. $\overline{D}_T$ denotes the expected ensemble diversity on correctly predicted examples, and $\overline{D}_F$ denotes the expected ensemble diversity on incorrectly predicted examples. $\overline{D}_T$, $\overline{D}_F$, and $\overline{D}$ will always be between 0 and 0.5. This can be easily seen for a two class dataset. If $\overline{D}$ became more than .5 than the ensemble would predict a new class thereby making $\overline{D}$ less than .5 again. The same argument holds for $\overline{D}_F$ and $\overline{D}_T$ and becomes even more obvious when there are more than two classes. The following lemmas hold:

Lemma 2. *Under 0-1 loss, the expected diversity* $\overline{D}$ *can be written as* $\overline{D} = (1 - L)\overline{D}_T + L\overline{D}_F$.

Lemma 3. *For 0-1 loss in two-class problems, the expected ensemble loss* L *can be written as*

$$L = \frac{\overline{L} - \overline{D}_T}{1 - \overline{D}_T - \overline{D}_F.}$$

Lemma 3 can be rewritten equivalently as

$$L = \overline{L} - (1 - L)\overline{D}_T + L\overline{D}_F. \tag{2}$$

Lemma 4. *For 0-1 loss in two-class problems, the expected ensemble loss L can be written as $L = \frac{\overline{L}+\overline{D}-2\overline{D}_T}{1-2\overline{D}_T}$.*

Lemma 5. *For 0-1 loss in two-class problems, the expected ensemble loss L can be written as $L = \frac{\overline{L}-\overline{D}}{1-2\overline{D}_F}$.*

As discussed in the following section, Lemmas 3, 4 and 5 provide better insight in the behavior of ensemble classifiers.

Relating L, $\overline{L}$, $\overline{D}$, $\overline{D}_T$ and $\overline{D}_F$: There are two ways to manipulate $\overline{D}_T$ and $\overline{D}_F$:

1. by keeping T and F constant. In this case, the ensemble decisions do not change, hence the 0-1 loss does not change, but the diversities $\overline{D}_T$ and $\overline{D}_F$ can be manipulated independently.
2. by changing the boundaries of T and F. In this case, the 0-1 loss is affected, and so are $\overline{D}_T$ and $\overline{D}_F$.

Case 1: When keeping T and F constant and increasing $\overline{D}_T$ by α, $\overline{D}$ increases by $\alpha(1 - L)$ (by Lemma 2) and $\overline{L}$ increases by $\alpha(1 - L)$ (because of Equation 2 and L remains constant). Lemma 4 shows that L remains constant under such change. If $\overline{D}_F$ is increased by α, $\overline{D}$ increases by αL (again by Lemma 2), and likewise $\overline{L}$ decreases by αL (because of Equation 2). Lemma 5 confirms that L remains constant under such change.

Case 2: Now consider manipulating T and F by moving instances from F to T such that L decreases by α. Let M be the set of instances $\{\langle \mathbf{x}, y\rangle \in X \times Y\}$ that moved from F to T. For those instances holds $\overline{L} - \overline{D} = 0$ by Lemma 2. So, by Lemma 4, we have $L - \alpha = (\overline{L} - \overline{D} - \alpha)/(1 - 2\overline{D}_F)$, which after some manipulations gives $\overline{D}_F = L - \overline{L} + \overline{D}/2(L - \alpha)$, that is, $\overline{D}_F$ increases. This also works the other way around; increasing $\overline{D}_F$ helps decrease the expected ensemble loss L.

Let us look at a case study to make this a bit clearer. In Figure 2 we have three instances in the set T and two instances in the set F. We have 3 member classifiers for classifying each of these instances; their votes are given beside each instance. In case 1 we increase $\overline{D}_T$ while keeping T and F constant; for example changing the votes at instance 1 from TTT to TTF. This will cause an increase in $\overline{D}$ and $\overline{L}$ but L remains constant. Likewise a change in instance 4, from FFF to FTF will cause an increase in $\overline{D}_F$, $\overline{D}$, but a decrease in $\overline{L}$ while L remains constant. This highlights that just increasing or decreasing diversity can have no effect on the benefits derived from an ensemble.

Now let us examine case 2. If we move instance 3 from the T set to the F set by changing its votes from FTT to FTF, we will cause L to increase while $\overline{D}$ hasn't changed. $\overline{L}$ will also increase and $\overline{D}_F$ could either have increased or decreased. If we move instance 5 from the F set to the T set by changing its votes

from FFT to FTT, we will cause L to decrease while $\overline{D}$ still hasn't changed. $\overline{L}$ will also decrease and $\overline{D}_T$ could either have increased or decreased. From this we can see, that increasing the diversity over the F set is the only way to lower the ensemble loss.

Why weighting may work: Instead of letting the classifiers in the ensemble vote uniformly ($\forall c \in C : w_c = 1/|C|$), the weights can be made proportional to the classification accuracy on the training data. This can be achieved by setting $\forall c \in C : w_c \propto 1 - L_c$, where $L_c = \int_{X \times Y} l(y, \hat{y}_c(\mathbf{x}))p(\langle \mathbf{x}, y \rangle)d\langle \mathbf{x}, y \rangle$ empirically estimates the expected loss of member classifier c on the training data, with m being the number of instances in the training set. We assume that when the classification accuracy on the training data is smaller, then the classification accuracy in general will be smaller, that is, performance on the training data can be seen as a predictor on the performance in general. This seems a reasonable assumption since it underlies all classifier learning algorithms (as long as overfitting is taken into account).

So, let's look at what happens when moving weight to classifiers in the ensemble that perform well on the training data. From the definition for expected loss $\overline{L}$ (definition 2) we see that it decreases. Also, the diversity $\overline{D}_F$ will decrease, hence from Lemma 5 we have that the expected loss will decrease (since the numerator decreases and the denominator increases).

Obviously the effect of increased performance by weighting is not the only mechanism at work. If this were the case, just taking the classifier in the ensemble with the highest accuracy on the training data and discarding the others would give a better classifier than using the whole ensemble in uniform voting. Experimental results suggest that this is generally not true (see [2] for details).

What diversity does for binary classes: Let's look at an example with a binary class, a set of four instances $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, and a set of three classifiers $C = \{c_1, c_2, c_3\}$, each classifier being able to classify two instances correctly and two incorrectly. For simplicity, a democratic voting scheme is used, so that it is easy to illustrate when classifiers are correct or incorrect. For the purpose of this example, it is also assumed that the weights are uniform ($w_c =$

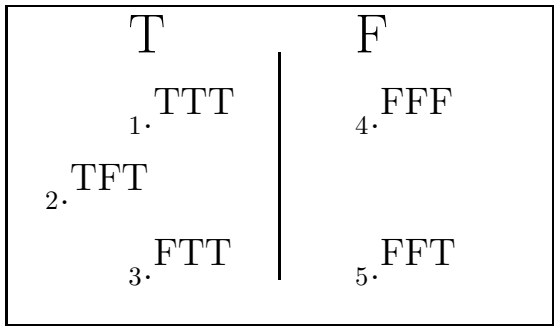

Fig. 2. Case study of relations between L, $\overline{L}$, $\overline{D}$ and y, $\widehat{y}$ and $\widehat{y}_c$.

1/3), that there is no class noise ($\forall i \in \{1,..,4\} : y_i = f(\mathbf{x}_i)$), and that the examples are uniformly distributed ($\forall \langle \mathbf{x}_i, y_i \rangle : P(\langle \mathbf{x}_i, y_i \rangle) = 1/4$).

The same effect appears when the set of classifiers is larger than 3. So, in general, we want to have a high as possible $\overline{D}_T$ with an as low as possible $\overline{D}_F$.

Probabilistic voting can be interpreted as democratic voting where, instead of a single vote by a single classifier c_i, a set of classifiers $c_{i,1}, \ldots, c_{i,|Y|}$ vote democratically, and a portion $\widehat{P}_c(y = 0)$ of those votes for $y = 0$ while the rest votes for $y = 1$. Therefore, the same effect can be observed in probabilistically voting ensembles as well.

Conclusions: A generalized unified decomposition of ensemble loss for predicting ensemble performance was presented that generalizes the unified decomposition presented in UDEL by allowing for classifier weights and probabilistic voting schemes. While UDEL was limited to uniform weights and democratic voting schemes, where the various base classifiers each can vote for a single class once only, this article extends the result to more general voting schemes. Experimental results suggest (see [2] for details) that democratic voting can be outperformed by probabilistic and progressive voting schemes, hence a GUDEL is worth exploring. Specifically probabilistic voting schemes are shown to provide a real advantage over democratic voting schemes in many datasets. So they should be tried more often in real world problem solving situations. In addition the formalization of ensembles presented gave insight into why the voting schemes provided such differing results.

Another class of classifiers for multinomial classes can be considered which calculates confidence on the probabilities of the classes through standard Bayesian technique of representing it as Dirichlet distribution. A voting scheme based on the confidence in the probabilities could be devised giving more voting rights to base classifiers with high confidence in their probabilities. The GUDEL can be extended to such voting schemes (although preliminary experiments suggest this may not be a fruitful route to pursue).

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: bagging boosting and variants. Machine learning, 36(1-2): 105-139, 1999.
2. R. Bouckaert, M. Goebel, P Riddle. Generalized Unified Decomposition of Ensemble Loss (full version of this paper). Available from http://www.cs.waikato.ac.nz/~remco/gudelfull.ps.
3. L. Breiman. Bagging predictors. Machine Learning 24(2): 123-140, 1996.
4. M. Goebel, P. Riddle, M. Barley. A unified decomposition of ensemble loss for predicting ensemble performance. ICML 2002.
5. Y. Singer and R. Schapire. Improved boosting algorithms using confidence-rated predictions. Machine Learning, 1999.