BRINGING DIVERSE CLASSIFIERS TO COMMON GROUNDS: DTRANSFORM

Devi Parikh and Tsuhan Chen

Carnegie Mellon University
Department of Electrical and Computer Engineering
{dparikh,tsuhan}@cmu.edu

ABSTRACT

Several classification scenarios employ multiple independently trained classifiers and the outputs of these classifiers need to be combined. However, since each of the trained classifiers exhibit different statistical characteristics, it is not appropriate to combine them using techniques that are blind to these differences. We propose a transform, *dtransform*, that transforms outputs of classifiers to approximate posterior probabilities, and caters to the statistical behavior of the classifier while doing so. The transformed outputs are now comparable, and can be combined using any of the classical combination rules. We show convincing results that demonstrate the effectiveness of the proposed transform in providing better estimates of the posterior probabilities as compared to standard transformations, as demonstrated by lower KL distance from the true distribution, higher classification accuracies and higher effectiveness of the standard classifier combination rules.

Index Terms— combining classifiers, estimating posterior probabilities, dtransform, parametric transformation of classifier outputs, intrusion detection

1. INTRODUCTION

Several scenarios utilize multiple independently trained classifiers. For instance, one-against-all classifiers are often trained for each class, and the class with the highest score is picked. Also, for robust classification, multiple weak classifiers are often trained to classify among the same classes, and their outputs are combined [1]. In both cases, it is important to have good estimates of the confidence of the classifiers, and more importantly, on the same *scale* so that they can be meaningfully compared/accumulated. Since the different classifiers can be of different types, or trained on different features, or trained on different subsets of the training data for diversity, or trained with learning algorithms that are not deterministic, and the underlying class conditional distributions themselves are varied, they are likely to have different statistical properties. And hence, combining their outputs without accounting for these differences is not appropriate.

We propose a transform, dtransform, which is classifier type independent, and transforms the outputs of the classifiers to estimates of posterior probabilities while incorporating these differences in statistical characteristics of the classifiers. The transformed outputs can now be combined using any of the standard classifier combination rules [1]. In this paper we focus on the transform itself, and not the classifier combination rules. dtransform is a parametric, non-linear transform that maps outputs of classifiers monotonically to estimates of posterior probabilities.

The rest of the paper is organized as follows. Section 2 discusses the motivation behind and contribution of dtransform compared to existing transforms, Section 3 describes dtransform, followed by experiments and results in Section 4 and lastly Section 5 concludes the paper.

2. CONTRIBUTIONS OF DTRANSFORM

We discuss two factors that strongly motivate the need for a transformation such as dtransform, and hence are the key contributions of dtransform as compared to other transforms. We note that we use the term 'multiple classifiers' loosely to not only refer to the conventional multiple classifiers systems, but also to include any multiple-class classification system that provides a score for each class, even if it is implemented as a single classifier with multiple outputs.

2.1. Adaptability

In most classification scenarios with multiple classes, the class corresponding to the maximum score (after scaling all to lie between 0 and 1 for instance) is taken to be the final classification decision. However, if the distribution of each classification output for the corresponding positive and negative class data points is observed, we would find that each one of them have a different optimum threshold, τ, that minimizes the classification error and can be determined according to Bayes classification rule [2].

For instance, consider an odor recognition system where a multi-layer perceptron (MLP) neural network with two outputs, d_1 and d_2, is trained to distinguish between two different odors, o_1 and o_2 respectively. If the output of d_1 is 0.6 and

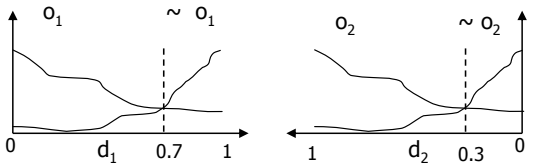

Fig. 1. Illustrative examples of the statistical characteristics of two outputs of an odor recognition system

that of d_2 is 0.4, most systems would detect the odor to be o_1 since it has a higher output. However, if we were to analyze the statistical characteristics of both detectors, we may find distributions as shown in Fig.1 for the outputs of the two detectors on validation data. According to the Bayes classification rule [2], d_1 has a value of $\tau_1 = 0.7$ and should pick o_1 only if its output is greater than 0.7, while τ_2 is 0.3. Hence, in the above case, the appropriate decision should be to pick o_2, and not o_1.

dtransform, since it incorporates the statistical properties of the detectors and adapts to them, would capture this, and would pick o_2 as the odor in the test instance. In two-class problems such as in biometric recognition systems such an analysis seems obvious and perhaps quite prevalent. However for most multi-class classification problems where classifiers such as multi-layer perceptron (MLP) neural networks or support vector machines (SVM) are trained for instance, such an analysis is often bypassed, and the threshold is assumed to be 0.5 for all the outputs of the MLP or 0 for each SVM, without analyzing the statistical behavior of each classifier/output post-training.

2.2. Non-linearity

In scenarios where multiple classifiers are to be combined, several classifier combination rules [1], require good estimates of posterior probabilities. Also, in scenarios where different classification errors have different costs associated with them, having good estimates of posterior probabilities would allow us to directly use Bayes classification rule with costs [2] to make optimum decisions.

Consider a scenario where, on a different test instance, d_1 gives an output of 0.8, and d_2 gives an output of 0.4. Which odor do we pick? Even if the above analysis is performed, and we realize that $\tau_1 = 0.7$ and $\tau_2 = 0.3$, if we merely look at the difference of the outputs from the corresponding thresholds, they are tied. But, it is intuitive that since τ_1 is much higher than τ_2, an increment of 0.1 in the output of d_1 may be considered to be significant, as compared to that of d_2. Hence, since each one of them have different τ's, a linear transform such as simply taking the difference of the output from its τ need not be appropriate. A good transform would map the τ of each classifiers to a probability of 0.5, and all other values should be mapped (monotonically) to the range of 0 and 1. This requires a non-linear transform, which dtransform provides. The other reason why this linear transform is not appropriate is because it could produce values outside the range of 0 and 1, which is not a valid distribution.

One could argue that the posterior probability can be estimated non-parametrically using something like Fig.1. However, this is prone to over-fitting and other violations of certain desired properties such as monotonic relationship between the output of the classifier and it's confidence. dtransform on the other hand estimates the distribution parametrically, learning only one parameter, the optimum threshold τ for each output.

Apart from ad hoc parameter tuning, principled approaches have been taken in trying to estimate posterior probability distributions from outputs of classifiers for several different classifier types. Duin *et al.* [3] use different methods for estimation for different types of classifiers, and hence is not classifier type independent. Moreover, the method does not incorporate the differences in statistical behavior of the individual classifiers post-training. dtransform is generic for most classifier types, and hence is more handy to employ, and more importantly, is specifically molded for each classifier. There has been work that indicates that outputs of MLPs (which we use in our experiments) provide estimates of posterior probabilities [4]. However, this being theoretical work, it does not address practical problems faced while training MLPs. To deal with these, most commonly used schemes are normalization [1] and softmax [5]. Again, none of these incorporate the statistical characteristics of the individual trained MLP, while dtransform does.

3. APPROACH

Having provided most of the philosophy and intuition behind dtransform in the previous section, we now describe the approach used to determine the transformation function. As stated earlier, given a classification system trained for multiple classes, the goal is to determine a transform for each output corresponding to each class. Since dtransform is parametric, this task boils down to determining the values of τ for each output.

Suppose there are C classes, and we have C classifier outputs, one corresponding to each class. Given a validation dataset, suppose μ_c is the score given by the c^{th} output to all the data points in the validation data set. Suppose the histogram of μ_c for the instances that belong to class c is $N_c^+(t)$ and the histogram of μ_c for the instances that do not belong to class c is given by $N_c^-(t)$. For a given value of threshold $t = \theta$, the number of misclassifications is given by

$$\kappa_c(\theta) = \sum_{t>\theta} N_c^-(t) + \sum_{t<\theta} N_c^+(t) \qquad (1)$$

The optimum threshold that minimizes the number of misclassification, according to Bayes classification rule [2], which can be computed by a simple 1D search, is

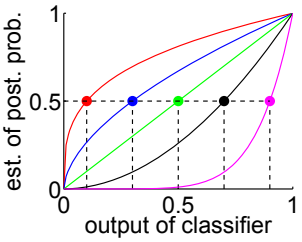

Fig. 2. The dtransform family of curves. From top to bottom, $\tau = 0.1, 0.3, 0.5, 0.7, 0.9$. Whenever the output of the classifier is τ, it is mapped to 0.5.

$$\tau_c = \underset{\theta}{\operatorname{argmin}}\, \kappa_c(\theta) \tag{2}$$

Now, in order for all outputs to be transformed such that they are now comparable, the τ for all outputs should be mapped to the same number. Since the transformed outputs are meant to be estimates of posterior probability, this number should be 0.5. We assume the outputs are scaled to lie between 0 and 1. Hence the following are the desired properties for the transform:

1. a raw output of 0 maps to a transformed output of 0
2. a raw output of 1 maps to a transformed output of 1
3. τ maps to 0.5
4. monotonically increasing

We pick the following form for dtransform that satisfies all the above properties. The family of curves corresponding to different values of τ is shown in Fig.2.

$$D(\mu;\tau) = \mu^{\frac{ln(0.5)}{ln(\tau)}} \tag{3}$$

where μ is the output score to be transformed, and τ, the optimum threshold determined for this classifier, is the parameter for dtransform.

One may argue that logistic regression [6], which has been used widely, for example to transform the outputs of Fisher Linear Discriminant classifier [3], could be used as the family of curves instead of dtransform. However, logistic regression does not provide the flexibility of properties (1) and (2), and more importantly, would require two parameters to be set, while dtransform has just one intuitive parameter.

4. EXPERIMENTS AND RESULTS

Experiments were performed to demonstrate the following as compared to other transforms: (1) Experiment 1: dtransform provides higher accuracies and lower KL distance from the true distribution (2) Experiment 2: dtransform makes classifier combination rules such as sum and product rules [1] more effective.

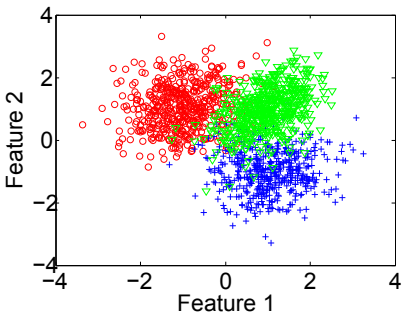

Fig. 3. Distribution of synthetic data points used

Table 1. Different transforms being compared

normalization	$n(\mu_i) = \frac{\mu_i}{\sum_{j=1}^{C} \mu_j}$
softmax	$s(\mu_i) = \frac{e^{\mu_i}}{\sum_{j=1}^{C} e^{\mu_j}}$
tsoftmax	$t(\mu_i) = \frac{e^{\mu_i - \tau_i}}{\sum_{j=1}^{C} e^{\mu_j - \tau_j}}$
dtransform	$d(\mu_i) = \frac{\mu_i^{\frac{ln(0.5)}{ln(\tau_i)}}}{\sum_{j=1}^{C} \mu_j^{\frac{ln(0.5)}{ln(\tau_j)}}}$

4.1. Higher accuracy and lower KL distance

We use synthetic data (2 features, 3 class) generated from gaussian distributions as shown in Fig.3, so that the ground truth posterior probabilities are known. 100 points were generated from each class, of which 30 were randomly selected for training, 30 for validation and the rest for testing.

An MLP was trained on the data, whose outputs were transformed using different transformations as listed in table 1 to get estimates of posterior probabilities. For each transform, the class corresponding to the output with the maximum value post-transformation was picked as the final classification decision. This experiment was repeated 100 times with random splits of the data. We compare the dtransform estimates to those obtained by normalization [1] and softmax [5] - none of which adapt the transformation based on the statistical characteristics of the classifier and hence do not incorporate τ, as well as and tsoftmax - which we design for comparison sake, and is similar to softmax, but takes τ into account. The forms of each of these are shown in Table 1, where we consider a C class classification problem, and the output scores for a given test instance are μ_i, $i \in 1, ..., C$. Each output has an optimum threshold τ_i associated with it. It can be seen that each of these are valid probability distributions with the values of the transformed outputs lying between 0 and 1 and the sum of all C outputs in each case being 1.

KL distance between the estimates and true posterior

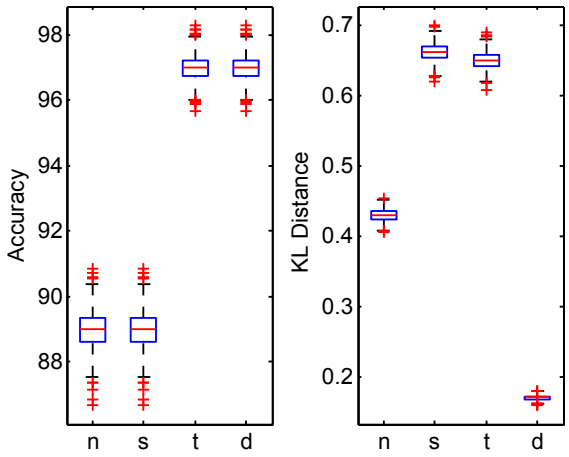

Fig. 4. Results on synthetic data using a single MLP neural network. Mean and 95% confidence intervals are shown. n = normalization, s = softmax, t = tsoftmax, d = dtransform

probability distributions, and the classification accuracy were used as the performance metrics. The results obtained can be seen in Fig.4. It can be seen that normalization and softmax both have the same (low) accuracies because they do not adapt according to τ, but normalization has a lower KL distance. dtransform has better classification accuracy because it adapts to each τ for every output. Since tsoftmax also incorporates the optimum threshold for each output, its classification accuracy is the same as that for dtransform, however its KL distance from the true distribution is high, which validates the parametric form dtransform uses. Hence, dtransform provides a higher classification accuracy as well as lower KL distance, while the other transforms lose out on at least one of these factors.

4.2. More effective classifier combination

We use a real intrusion detection dataset from KDD 1999 [7] which has over 5 million data points, 41 features coming from 3 different sources of information, and 5 classes - 1 corresponding to normal traffic and 4 corresponding to different attack types. A 5×5 cost matrix is also associated with the different classification errors, and is provided with the dataset. An ensemble of classifiers approach capable of data fusion, inspired in part by Adaboost, Learn++, was used. It uses hierarchical ensembles of classifiers and combines information from multiple sources. In order to do so, it employs several sum and product rules [1] to combine the multiple classifiers. Details can be found in [8]. We use MLP neural networks as the base classifiers. dtransform was applied to the outputs of all MLPs before combining them at various stages in Learn++. The cost matrix was considered using the Bayes classification rule for cost [2]. Fig.5 shows that the cost per instance incurred using dtransform is lower than that obtained

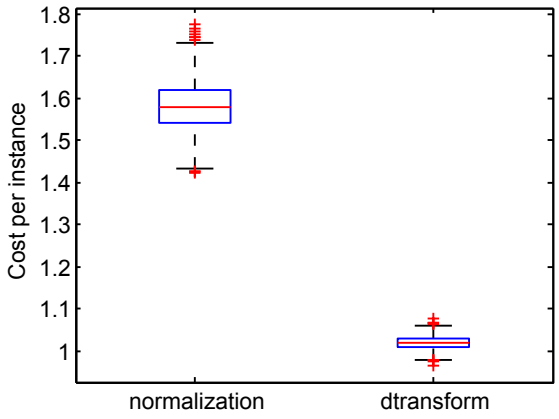

Fig. 5. Results on a real intrusion detection dataset using ensembles of classifiers. Mean and 95% confidence intervals are shown.

without employing dtransform (using just normalization) with statistical significance. This shows that dtransform makes the classifier combination rules more effective, while making the cost considerations straightforward.

5. CONCLUSION

We proposed dtransform, a simple yet principled, parametric, non-linear transformation of the outputs of classifiers to estimates of posterior probabilities. It is classifier type independent, however it incorporates the statistical characteristics of each classifier so that the classifiers are now on fair grounds to be compared/aggregated. We demonstrated the effectiveness of dtransform in providing better estimates of posterior probabilities than standard transformations.

6. REFERENCES

[1] J. Kittler, M. Hatef, R. Duin and J. Matas. On combining classifiers. In *PAMI*, 1998.

[2] D. Duda, P. Hart and D. Stork. In Pattern Classification, 2/e, Chap. 2, pp. 20-29, New York, NY: Wiley Interscience, 2001.

[3] R. Duin and D. Tax. Classifier conditional posterior probabilities. In *LNCS*, 1998.

[4] M. Richard and R. Lippmann. Neural network classifiers estimate Bayesian a-posteriori probabilities. In *Neural Computation*, 1991.

[5] E. Alpaydin and M. Jordan. Local linear perceptrons for classification. In *IEEE Trans. on Neural Networks*, 1996.

[6] A. Anderson. Logistic discrimination. In P. Krishnaiah and L. Kanal (eds.), Handbook of Statistics 2: Classification, Pattern Recognition and Reduction of Dimensionality, 1982.

[7] The UCI KDD Archive, Information and Computer Science, University of California, Irvine, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[8] M. Lewitt and R. Polikar. An ensemble approach for data fusion with Learn++. In *Proc. Intl. Wrk. on Multiple Classifier Systems (MCS)*, 2003.