Combining Linear Dichomotizers to Construct Nonlinear Polychotomizers

Ethem Alpaydın [a]   Eddy Mayoraz [b]

IDIAP–RR 98-05

May 1998

[a] Visiting scholar at IDIAP while this work was done. On sabbatical leave from the Department of Computer Engineering, Boğaziçi University, TR-80815 Istanbul Turkey. alpaydin@boun.edu.tr
[b] IDIAP, CP 592, CH-1920 Martigny Switzerland. mayoraz@idiap.ch

IDIAP Research Report 98-05

Combining Linear Dichomotizers to Construct Nonlinear Polychotomizers

Ethem Alpaydın Eddy Mayoraz

May 1998

Abstract. A polychotomizer which assigns the input to one of $K, K \geq 3$, is constructed using a set of dichotomizers which assign the input to one of two classes. We propose techniques to construct a set of linear dichotomizers whose combined decision forms a nonlinear polychotomizer, to extract structure from data. One way is using error-correcting output codes (ECOC). We propose to incorporate soft weight sharing in training a multilayer perceptron (MLP) to force the second layer weights to a bimodal distribution to be able to interpret them as the decomposition matrix of classes in terms of dichotomizers. This technique can also be used to finetune a set of dichotomizers already generated, for example using ECOC; in such a case, ECOC defines the target values for hidden units in an MLP, facilitating training. Simulation results on eight datasets indicate that compared with a linear one-per-class polychotomizer, pairwise linear dichotomizers and ECOC-based linear dichotomizers, this method generates more accurate classifiers. We also propose and test a method of incremental construction whereby the required number of dichotomizers is determined automatically as opposed to assumed a priori.

1 Introduction

Classification is the assignment of a multidimensional input $\boldsymbol{x}$ to one of K classes, $\mathcal{C}_i, i = 1, \ldots, K$. For $K = 2$, the system is a *dichotomizer*. For $K \geq 3$, it is called a *polychotomizer*. It is generally the case that separating a class from all other classes may be difficult and requires a complex discriminant function which for good generalization needs a large training sample. The alternative is to define a set of simpler classifiers, each specializing on one aspect thereby solving a simpler problem. Combining these simpler classifiers we get the final classifier.

For each dichotomizer $a_l, l = 1, \ldots, L$, we define a sample of positive examples $\mathcal{X}_l^+$ and a sample of negative examples $\mathcal{X}_l^-$. a_l is trained to give $+1$ for $\boldsymbol{x} \in \mathcal{X}_l^+$ and -1 for $\boldsymbol{x} \in \mathcal{X}_l^-$. The *decomposition matrix* $D = [d_{kl}]$ of size $K \times L$ associates classes $\mathcal{C}_k, k = 1, \ldots, K$, to the samples $\mathcal{X}_l^+$ and $\mathcal{X}_l^-$:

$$d_{kl} = \begin{cases} +1 & \text{means } \mathcal{C}_k \subset \mathcal{X}_l^+ \\ -1 & \text{means } \mathcal{C}_k \subset \mathcal{X}_l^- \\ 0 & \text{means } \mathcal{C}_k \cap (\mathcal{X}_l^+ \cup \mathcal{X}_l^-) = \emptyset \end{cases} \tag{1}$$

Rows of D correspond to the definition of a class as a vector of responses of the L dichotomizers. For example with four dichotomizers, if row k is $[-1, +1, 0, +1]$, for us to say the input belongs to $\mathcal{C}_k$, we should have $a_1 = -1, a_2 = +1, a_4 = +1$ and we do not care for the value of a_3. Whereas if column l is $[+1, -1, +1]^T$ then we see that a_l should separate $\mathcal{C}_2$ from $\mathcal{C}_1$ and $\mathcal{C}_3$.

Once all dichotomizers are trained, given a pattern to classify, all a_l compute their outputs and we assign the pattern to the class having the closest representation (row of D). When the vectors are $-1/+1$, this can be done by taking a dot product

$$c = \arg\max_k \sum_l a_l d_{kl} \tag{2}$$

The usual *one-per-class* classification scheme corresponds to using a square ($L = K$) matrix D where $d_{kk} = +1$ and $d_{kl} = -1, k \neq l$ otherwise. That is, $\mathcal{X}_k^+ = \{\boldsymbol{x} | \boldsymbol{x} \in \mathcal{C}_k\}$ and $\mathcal{X}_k^- = \{\boldsymbol{x} | \boldsymbol{x} \in \mathcal{C}_l, l \neq k\}$.

2 Pairwise Classifiers

To get tolerance to faults (failure of a_l), we add redundance by defining additional dichotomizers. One immediate possibility is to build $K(K-1)/2$ *pairwise classifiers*, each being trained to separate examples of one class from examples of one other class, not using the examples of other classes [3]. Matrix D is of size $K \times K(K-1)/2$. For example for $K = 4$, we have

$$D = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

One problem with pairwise classification is that the number of dichotomizers is $\mathcal{O}(K^2)$ which may be expensive when K is large. Another is that each dichotomizer is trained with data from a smaller training set of only two classes and thus has higher variance. Related to this fact is that the response of dichotomizer separating $\mathcal{C}_i$ from $\mathcal{C}_j$ gives no information at all when the example does not belong to $\mathcal{C}_i$ or $\mathcal{C}_j$ [6]. To alleviate these deficiencies, a smaller number of dichotomizers which are all trained with the whole dataset is desirable.

3 Error Correcting Output Code Dichotomizers

To get robustness in case the dichotomizers fail, we should have rows of D as different as possible in terms of Hamming distance by adding redundancy; this is the idea of *error correcting output codes*

(ECOC) [1] and is used to decompose polychotomies into dichotomies [4, 6]. With K classes, we have up to $2^{(K-1)}-1$ dichotomizers. For example, with four classes the D matrix is

$$D = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{bmatrix}$$

Dietterich and Bakiri [1] report that with C4.5 trees, the dichotomizers trained to learn the ECOC dichotomizers are larger than the one-per-class tree and when multilayer perceptrons are used instead of trees, there may be a problem of convergence. These imply that the tasks defined by the columns of D may be more difficult than the original problem of separating one class from all others.

4 Learning the Decomposition Matrix

Let us take a multilayer perceptron with L sigmoidal hidden units and softmax outputs for classification

$$a_l = \tanh(\boldsymbol{w}_l^T \boldsymbol{x}) \ , \ \mu_k = \frac{\exp[\sum_{l=0}^{L} d_{kl} a_l]}{\sum_{i=1}^{K} \exp[\sum_{l=0}^{L} d_{kl} a_l]} \qquad (3)$$

with bias units added, i.e., $x_0 = a_0 = 1$, trained to minimize the cross-entropy on a dataset $\mathcal{X} = \{\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}\}_t$

$$E(\mathcal{X}|w_l, d_{kl}) = -\sum_t \sum_k y_k^{(t)} \log \mu_k^{(t)} \qquad (4)$$

In this structure, each sigmoidal hidden unit can be seen as a dichotomizer (in the range $[-1,+1]$) and the second layer weights d_{kl} constitute the decomposition matrix D. The advantage of this structure is that, when compared with pairwise classification, it may be trained with less than $K \times (K-1)/2$ dichotomizers and all dichotomizers are trained with the whole sample. Compared with ECOC, the partitioning of classes is not done a priori but is coupled with the training of the dichotomizers such that any deficiency can be corrected by other dichotomizers.

As it is, there is no restriction on d_{kl} values. If we want to get a D with binary entries as in Eq. 1, we can force this through *soft weight sharing* [7] by assuming that d_{kl} (nonbias weights only) are sampled from a bimodal density, e.g., mixture of two Gaussians

$$p(d) = P(1)p(d|1) + P(2)p(d|2) = \alpha_1 p_1(d) + \alpha_2 p_2(d) \qquad (5)$$

$\alpha_m, m = 1, 2$, are priors and p_m are Gaussian components $p(d|m) \equiv p_m(d) \sim \mathcal{N}(c_m, s_m^2)$ with $c_1 = -1$ and $c_2 = +1$. Maximizing the likelihood of d_{kl} is equivalent to adding the following regularization term to the cross-entropy

$$\Omega = -\nu \sum_k \sum_l \log\left[\alpha_1 p_1(d_{kl}) + \alpha_2 p_2(d_{kl})\right] \qquad (6)$$

Note that this is a Maximum A Posteriori (MAP) approach where cross-entropy is the $-\log$ of the likelihood (Eq. 4) and Ω is the $-\log$ of the prior (Eq. 6 with ν as a common factor in the component variances); thus their sum is the $-\log$ of the posterior. Using gradient-descent, we can get update equations for d_{kl} as well as α_m and s_m^2 (after adding auxiliary variables to guarantee that variances are nonnegative and priors sum up to one).

A way that is simpler to implement than defining a prior is to have a set of auxiliary variables z_{kl} and define

$$d_{kl} = \tanh(\beta z_{kl}) \qquad (7)$$

where β defines the sharpness. We can learn z_{kl} by backpropagating to minimize Eq. (4). tanh forces d_{kl} to be in the interval $[-1,+1]$ saturating in the two extremes where the derivative is zero. This is similar to having two Gaussians where the posterior, $\alpha_m p_m(d)/[\alpha_1 p_1(d) + \alpha_2 p_2(d)]$, is a sigmoid, which has the same shape as tanh. Smaller s_m^2 correspond to sharper sigmoids, i.e., larger β.

5 Incremental Construction of Dichotomizers

We can also add dichotomizers incrementally as opposed to assuming a fixed number a priori. At each iteration, if we do not have yet enough accuracy on the training set, we add a dichotomizer such that it separates classes which we currently have the largest difficulty discriminating. The algorithm is as follows:

1. $L \leftarrow 1$. Choose two classes at random, i and j.
2. $D_{iL} \leftarrow +1, D_{jL} \leftarrow -1, D_{kL} \leftarrow 0, \forall k, k \neq i, k \neq j$
3. Train a_L according to the definition in column L of D.
4. $\forall k$ such that $D_{kL} = 0$, count p_k^+, examples belonging to class k such that $a_L \geq 0$. Similarly count p_k^-, examples belonging to class k such that $a_L < 0$. Normalize by dividing with priors, $P(k)$.
5. Choose class g that is best separated by a_l, measured as minimum entropy
$$g \leftarrow \arg\min_k \left[-p_k^+ \log p_k^+ - p_k^- \log p_k^-\right]$$
Note that entropy is close to 0 if p_k^+ or p_k^- is close to 1, that is if all examples of $\mathcal{C}_k$ are on the positive or negative side of the hyperplane.
6. If $p_g^+ > p_g^-$, $D_{gL} \leftarrow +1$, otherwise $D_{gL} \leftarrow -1$. If $\exists k$ s.t. $D_{kL} = 0$, goto (3). If the new row, or its complement, already exists, do not add, goto (9).
7. Using dichotomizers $a_l, l = 1, \ldots, L$ trained up to this point and matrix D that is $K \times L$, classify points in the training set.
8. If sufficient accuracy, as given by a preset threshold, is attained or if $L = L_{max}$, stop.
9. Compute confusion matrix C where C_{ij} is the number of examples belonging to $\mathcal{C}_i$ but are assigned to $\mathcal{C}_j$. The two classes which we have most difficulty separating are the coordinates of the max entry in C
$$i, j \leftarrow \arg\max_{g,k} [C_{gk} + C_{kg}]$$
$L \leftarrow L + 1$. Goto (2)

This structure has the advantage that we can determine automatically the number of dichotomizers necessary as a function of the required accuracy. Note that with linear dichotomizers, we cannot get arbitrarily high accuracy, as the columns of D may correspond to tasks that are not linearly separable. This is a variant of the *Pertinent-ECOC* algorithm [4] where in Step (9), we choose the two classes whose definition in matrix D (rows of D) has the least Hamming distance.

6 Simulation Results

We compare the techniques we discussed on several datasets. `digsma` is by I. Guyon and contains 16×16 bitmaps of ten handwritten digits. `digit` and `pen` are respectively for optical and pen-based handwritten digit recognition and are available through the authors. The other datasets are from UCI repository [5].

We first compare the linear one-per-class polychotomizer and pairwise linear dichotomizers (Table 1). We use the 5x2cv t test [2] as the test of statistically significant difference, except on `letter` where only one run is done and the McNemar test [2] is used. In five of the eight datasets, the accuracy of pairwise dichotomizers is significantly superior to that of the one-per-class polychotomizer; these are the datasets where there is a difference between having $K(K-1)/2$ and K linear models.

The pairwise classifier in turn is compared with the MLP-Soft started from random initial weights. In Figure 1, we see that with the constraints added, the second layer weights of an MLP do converge

Table 1: Comparison of linear one-per-class polychotomizer, pairwise linear dichotomizers, MLP-Soft and normal MLP. Values are L: average, stdev accuracies of ten independent runs on the test set, except on `letter` where only one run is made. '*' indicates statistically significantly (0.95) better than the previous column. '<' means worse.

	One-per-Class	Pairwise		MLP-Soft		vanilla MLP	
`glass`	7: 61.87, 3.94	21: 61.87, 3.55		21: 66.82, 4.18		66.45, 4.64	
`vowel`	11: 44.18, 0.99	55: 66.38, 2.48	*	55: 83.90, 1.81	*	76.89, 2.00	<
`thyroid`	3: 95.45, 0.33	3: 95.11, 0.35		3: 94.27, 0.47		97.75, 0.89	*
`satimage`	7: 85.37, 0.26	21: 86.36, 0.33	*	21: 88.40, 0.35	*	88.80, 0.50	*
`letter`	26: 76.39	325: 77.93	*	100: 87.98	*	88.38	
`digsma`	10: 95.27, 0.75	45: 97.48, 0.82		45: 97.83, 0.51		98.22, 0.53	
`digit`	10: 96.16, 0.50	45: 97.62, 0.21	*	45: 97.32, 0.32		97.13, 0.40	
`pen`	10: 90.95, 1.44	45: 94.62, 0.41	*	45: 96.52, 0.85	*	98.63, 0.12	

to a bimodal distribution without losing accuracy. We notice that the approach given in Eq. (7) where auxiliary parameters are filtered through tanh converge better than soft weight sharing with two Gaussians. We name this MLP-Soft which in four datasets is significantly more accurate than pairwise classification. When K is large, a pairwise scheme may be too costly (e.g., on `letter`). MLP-Soft is able to learn with arbitrary L and its accuracy is never less than normal MLP without constraints (Table 1) —MAP vs ML estimation.

MLP-Soft can also be used to improve the accuracy of a given set of dichotomizers as opposed to training from scratch. Though the pure ECOC assumes strong learners that can learn the defined dichotomizers and thus fails with linear dichotomizers because the tasks are generally not linearly separable, by transforming the structure into an MLP and then finetuneing the dichotomizers and the decomposition matrix, the accuracy of ECOC is improved. That is the dichotomizers generated by ECOC define the first layer weights and we take an approximate inverse of tanh to initialize z_{kl}

$$z_{kl} = \begin{cases} +1.5 & \text{if } d_{kl} = +1 \\ -1.5 & \text{if } d_{kl} = -1 \end{cases} \tag{8}$$

We take values off the saturation regions for the derivative to be nonzero. Having constructed a bona fide MLP, we can finetune the dichotomizers and the decomposition matrix by training them in a coupled manner as the first and second layer weights. During finetuning, we use a small learning factor to keep the structure generated by ECOC. As shown in Table 2, in four of the eight datasets, this significantly improves accuracy. The same can also be applied to incremental method given in Section 5 leading to more accurate classifiers after finetuning.

This approach is interesting in that ECOC (or the incremental method) defines a target value for the hidden units of an MLP in a classification problem. This allows dividing the task of training a two-layer perceptron into the parallel training of several one-layer perceptrons for which faster convergence can be attained without sacrificing from accuracy (Table 2).

7 Conclusions

We propose techniques to construct a set of linear dichotomizers whose combined decision forms a nonlinear polychotomizer. This allows breaking a complex problem into a set of simpler problems permitting extraction of structured knowledge from data. We show how, by adding constraints, the backprop algorithm realizes this during gradient-descent, without losing from accuracy when compared with backprop proper. Our second contribution is to use ECOC or an incremental method to define the target values for the hidden units of an MLP. This leads to faster convergence as training a two-layer perceptron is converted to the parallel training of several one-layer perceptrons, without any sacrifice from accuracy.

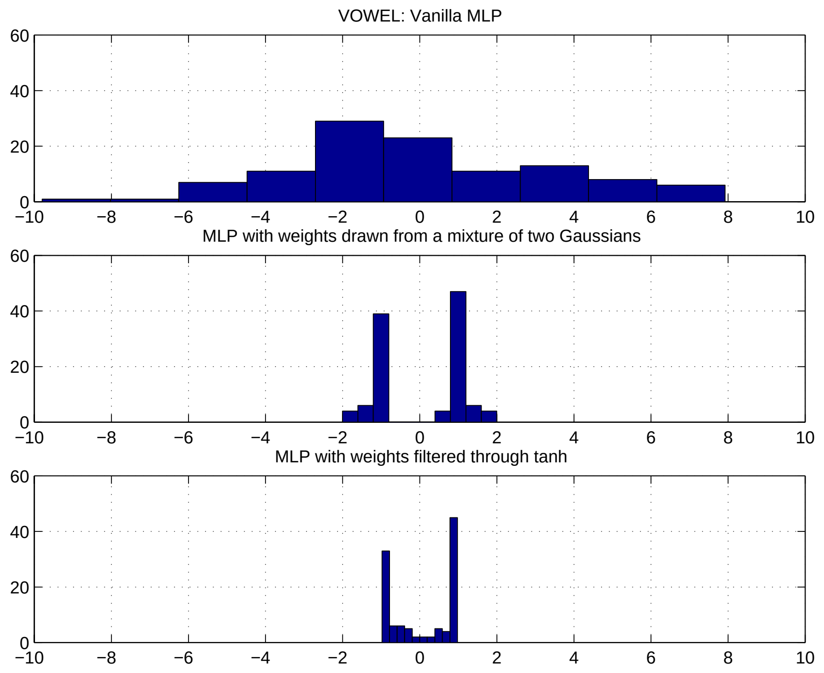

Figure 1: For one run on `vowel` with 10 hidden units/dichotomizers, the histograms of weights are given. (a) Second layer weights of a normal MLP, (b) Second layer weights of an MLP where weights are drawn from a bimodal Gaussian with centers at -1 and 1, (c) Like (b) except that the weights are filtered through tanh with extremes at -1 and $+1$.

Table 2: Accuracy with ECOC-linear and the incremental method; raw and after finetuning. '*' indicates statistically significant improvement during finetuning. Accuracy of MLP-Soft with the same L starting from random weights is also given.

	ECOC-linear			Incremental construction			
	L: Raw	Finetuned	?	L: Raw	Finetuned	?	MLP-Soft
`glass`	7: 58.88, 3.29	67.00, 3.74		6: 65.00, 2.00	67.50, 2.12		64.00, 0.67
`vowel`	11: 24.76, 4.66	47.71, 4.22		10: 40.84, 3.79	45.50, 3.99	*	53.16, 1.82
`thyroid`	3: 94.69, 0.38	94.78, 0.06		2: 93.29, 0.28	93.83, 0.08		93.09, 0.37
`satimage`	7: 81.00, 2.82	88.16, 0.49	*	8: 81.28, 0.87	87.89, 0.49	*	85.87, 0.51
`letter`	26: 39.84	89.43	*	30: 54.68	80.64	*	81.06
`digsma`	20: 90.82, 0.72	90.57, 1.40		20: 85.17, 2.87	95.50, 0.52	*	97.35, 0.36
`digit`	20: 91.19, 0.90	93.21, 0.46	*	20: 85.42, 1.34	93.88, 0.37	*	95.11, 0.27
`pen`	20: 84.25, 1.23	94.91, 0.60	*	20: 76.76, 1.78	95.40, 0.38	*	95.35, 0.15

References

[1] T. G. Dietterich, G. Bakiri (1995). "Solving Multiclass Learning Programs via Error-Correcting Output Codes," *Journal of Artificial Intelligence Research*, **2**, 263–286.

[2] T. G. Dietterich (1998). "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms," *Neural Computation*, to appear.

[3] R. O. Duda, P. E. Hart (1973). *Pattern Classification Scene Analysis*, Wiley.

[4] E. Mayoraz, M. Moreira (1997). "On the Decomposition of Polychotomies into Dichotomies", *14th Int'l Conf on Machine Learning*, Nashville, TN, 219–226.

[5] C. J. Merz, P. M. Murphy (1998). UCI Repository of Machine Learning Databases. `http://www.ics.uci.edu/ mlearn/MLRepository.html`.

[6] M. Moreira, E. Mayoraz (1998). "Improved Pairwise Coupling Classification with Correcting Classifiers", *Proc 10th European Conf on Machine Learning*, Chemnitz, Germany, April, 160–171.

[7] S. Nowlan, G. Hinton (1992). "Simplifying Neural Networks by Soft Weight Sharing," *Neural Computation*, **4**, 473–493.