Parallel Perceptrons, Activation Margins and Imbalanced Training Set Pruning

Iván Cantador and José R. Dorronsoro *

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid, 28049 Madrid, Spain

Abstract. A natural way to deal with training samples in imbalanced class problems is to prune them removing redundant patterns, easy to classify and probably over represented, and label noisy patterns that belonging to one class are labelled as members of another. This allows classifier construction to focus on borderline patterns, likely to be the most informative ones. To appropriately define the above subsets, in this work we will use as base classifiers the so–called parallel perceptrons, a novel approach to committee machine training that allows, among other things, to naturally define margins for hidden unit activations. We shall use these margins to define the above pattern types and to iteratively perform subsample selections in an initial training set that enhance classification accuracy and allow for a balanced classifier performance even when class sizes are greatly different.

1 Introduction

Most real world classification problems involve imbalanced samples, that is, samples where the number of patterns from one class (that we term the positive samples) is much smaller than that from others. There are many examples of this situation [5, 4], as well as a large literature on this topic, for which many techniques have been applied. Basic examples are ROC curves [8, 12] or the alteration of the sample class distribution, either by oversampling the minority class [2], undersampling the majority class [7] or doing this on both [3]. Moreover, sampling techniques are also in the core of the more sophisticated methods that arise from the well known boosting paradigm [6].

In this work we shall propose a new procedure for training set reduction based on the concept of margin that arises naturally in parallel perceptron (PP) training introduced by Auer et al. in [1]. Parallel perceptrons have the same structure of the well known committee machines [10], that is, they are made up of an odd number of standard perceptrons P_i with ± 1 outputs, and the machine's one dimensional output is simply the sum of these perceptrons' outputs (that is, the overall perceptron vote count). They are thus well suited for 2–class discrimination problems, but it is shown in [1] that they can also be used in regression problems, as they have indeed a universal approximation property.

* With partial support of Spain's CICyT, TIC 01–572, TIN2004–07676.

Another contribution of [1] is to give a general and effective training procedure for PPs. A key part of this training procedure is a margin based output stabilization technique that tries to augment the distance of the activation of a perceptron from its decision hyperplane, so that small random changes on an input pattern do not cause its being assigned to another class. Although these margins are not defined on the one dimensional output of a PP and they have to be considered independently for each perceptron, they do provide a way to measure the relevance of individual patterns with respect the overall training set and to establish a pattern selection strategy.

We shall briefly describe in section 2 the training of PPs, as well as their handling of margins, while in section 3 we will describe the overall training set selection procedure and shall also see how margins can be used to discard both redundant patterns, that is, those patterns easy to detect and well represented by other patterns in the training set, and label noisy patterns, that is, those labelled as belonging to one class while their features clearly establish them as a member of another, while allowing to retain those patterns most interesting for training purposes. In section 4 we will illustrate numerically the results provided by the pattern selection algorithm over 7 example databases obtained fom the UCI repository. As we shall see, in all of them we arrive at much smaller training subsets that nevertheless allow the construction of effective PP classifiers. The paper ends with a brief summary section.

2 Parallel perceptron training

The parallel perceptron architecture is simply that of the well known committee machines. Let us briefly review it. Assume we are working with D dimensional patterns $X = (x_1, \ldots, x_D)^t$, where the D–th entry has a fixed 1 value to include bias effects. If the committee machine (CM) has H perceptrons, each with a weight vector W_i, for a given input X, the output of perceptron i is then $P_i(X) = s(W_i \cdot X) = s(act_i(X))$, where $s(\cdot)$ denotes the sign function and $act_i(X) = W_i \cdot X$ is the activation of perceptron i due to X. We then have

$$\sum_{1}^{H} P_i(X) = \#\{i : W_i \cdot X > 0\} - \#\{i : W_i \cdot X < 0\} = N_+(X) - N_-(X) = \mathcal{N}(X),$$

and the output $h(X)$ of the CM is $h(X) = s\left(\mathcal{N}(X)\right)$ where we take H to be odd to avoid ties. We will assume that each input X has an associated ± 1 label l_X and take the output $h(X)$ as correct if $l_X h(X) > 0$. It is then clear that X has been correctly classified if either $N_+(X) > N_-(X)$ when $l_X = 1$ or $N_+(X) < N_-(X)$ when $l_X = -1$. If this is not the case, and we have, say, $l_X = 1$, then $N_-(X) = (H - \mathcal{N}(X))/2$. Classical CM training ([10], ch. 6) then tries to change the smallest number of perceptron outputs so that X could then be correctly classified, and it is easy to see that this number is $(1 + |\mathcal{N}(X)|)/2$; this last formula can also be applied to wrongly classified X such that $l_X = -1$. Then,

whenever $l_X h(X) = -1$, classical CM training first selects those $(1 + |\mathcal{N}(X)|)/2$ perceptrons P_i such that $l_X P_i(X) = -1$ and for which $|act_i(X)|$ is smallest, and changes their weights by the well known Rosenblatt's rule:

$$W_i := W_i + \eta l_X X. \tag{1}$$

CMs and parallel perceptrons (PPs) differ in their training. PPs can be trained either on line or, as we shall do here, in batch mode and for them the update (1) is applied to all wrong perceptrons, i.e. those P_i verifying $l_X P_i(X) = -1$. Moreover, their training has a second ingredient, a margin–based output stabilization procedure. Notice that if $W_i \cdot X \simeq 0$, small changes on X may cause a wrong class assignment for a small perturbation of X. To avoid this instability, the update (1) is also applied when a pattern X is correctly classified but still $0 < l_X Act_i(X) < \gamma$.

The value of the margin γ is also adjusted dynamically from a starting value. More precisely, after a pattern X is processed correctly, we have

$$\gamma := \gamma + \eta \left(M_{min} - \min\{M_{max}, M(X)\} \right),$$

where $M(X)$ is the number of hyperplanes that process X correctly although with a too small margin. In other words, $M(X) = \#\{i : 0 < l_X Act_i(X) < \gamma\}$ for those X such that $l_X f(X) > 0$. Values proposed in [1] for M_{min} and M_{max} are $M_{min} = 0.25$ and $M_{max} = 1$. Observe then that γ increases if all correct perceptron activations are above the current margin (for then $M(X) = 0$), while it decreases if at least one perceptron activation is "below" the current margin (then $M(X) \geq 1$). Notice that for the margin to be meaningful, weights have to be normalized somehow; we will make its euclidean norm to be 1 after each batch pass. Notice PPs provide a common margin value γ for all H perceptrons; however, not all patterns have to behave with respect to γ in the same way overall H perceptrons.

In spite of their very simple structure, PPs do have a universal approximation property. Moreover, as shown in [1], PPs provide results in classification and regression problems quite close to those offered by procedures such as MLPs and C4.5 decision trees. Finally, their training is very fast, because of the very simple update (1) and because it is only applied to patterns incorrectly classified. Denoting their number as N_W and omitting for simplicity updates due to margins, the overall training complexity for a PP is $O(N_W DH)$; as training advances, we should have $N_W \ll N$ and, hence, very fast bacth iterations.

3 Training pattern selection

When dealing with imbalanced data sets, it is reasonable to expect patterns to fall within three cathegories, redundant, label noisy and borderline. Label noisy are simply those X for which their label assignment is likely to be wrong. It is thus desirable to exclude them from the training set. Redundant patterns are those easy to classify. Since they are likely to be overrepresented on the training

set, many of them can be possibly ignored during training without hampering classifier construction. Finally, borderline patterns are those whose classification could be different after small perturbations and therefore, classifier construction should concentrate on them to provide stable and possibly correct classifications after training ends.

This training pattern cathegorization can be potentially quite useful, for once achieved, classifier construction can proceed by iteratively constructing a sequence of classifiers using training sets where redundant and noisy patterns are progressively removed. Notice that this training can be viewed as a kind of radical boosting–like procedure, where redundant and noisy patterns probabilities change to 0 after each iteration while the remaining patterns are taken as equiprobable. The difficulty obviously lies on how to characterize patterns beloging to each class. For this, activation margins are a natural choice. Recall that PPs adaptatively adjust this margin, making it to converge to a final value γ. If for a pattern X its i–th perceptron activation verifies $|act_i(X)| > \gamma$, it is likely to remain so after a small perturbation. Thus if for all i we have $l_X act_i(X) > \gamma$, X is likely to be also correctly classified later on. Those patterns are natural choices to be taken as redundant. Similarly, if for all i we have $l_X act_i(X) < -\gamma$, X is likely to remain wrongly classified, and we will take such patterns as label noisy. The remanining X will be the borderline patterns. We shall use the notations R_i, N_i and B_i for the redundant, noisy and borderline training sets at iteration i. With a slight abuse of the language, we shall call a pattern's normalized activation $l_X act_i(X)$ its "margin".

After iteration i we shall remove the R_i and N_i subsets. With respect to B_i the first option is to keep all of its patterns after each iteration. However, working with imbalanced data sets we should treat positive and negative training patterns in different ways, specially if classes are mixed. The alternative option we shall also use is to remove after each iteration the subset nB_i^- of "noisy" borderline negative patterns X with a wrong margin $l_X act_i(X) < 0$ in all perceptrons. Although we could also do the same for the noisy borderline positive set nB_i^+, this has the risk of removing a sizeable amount of the positive patterns, whose number could be much smaller than that of the negative ones. This second alternative option certainly favors the positive class, so it has to be balanced somehow. We shall do so with the termination criterium to be used. Recall that we want a good classification performance on the positive class, but maintaining also a good performance on the negative class. We thus need to balance the positive and negative accuracies, defined as $a^+ = TP/(TP + FN)$ and $a^- = TN/(TN + FP)$, where TP, TN denote the number of true positives and negatives, that is, positive and negative patterns correctly classified, and FP, FN denote the number of false positives and negatives, that is, negative and positive patterns incorrectly classified. For imbalanced problems, simple accuracy, that is the percentage of correctly classified patterns, may not be a relevant criterium, as it would be fulfilled by the simple (and very uninteresting) procedure of assigning all patterns to the (possibly much larger) negative classes. Other criteria are thus needed, and several options such as precision (the ratio

Problem set	% positives	final g with nB^-	final g without nB^-	MLP–BP
cancer	34.5	96.5	96.6	96.1
diabetes	34.9	68.7	71.2	71.7
ionosphere	35.9	77.6	82.3	80.8
vehicle	25.7	69.6	72.5	75.7
glass	13.6	91.2	91.6	91.8
vowel	9.1	92.9	93.3	97.1
thyroid	7.4	73.9	97.2	95.8

Table 1. The table gives final g values when nB^- is kept (third column) and removed (fourth); the second option gives better results. For comparison purposes g values obtained after direct MLP are also given.

$TP/(TP+FP)$) or recall (the ratio $TP/(TP+FN)$, i.e., our positive class accuracy a^+), ROC curves, or other, have been proposed. Here we shall use a simple measure first used in [11], the geometric ratio $g = \sqrt{a^+ a^-}$ between positive and negative accuracies, that measures the balance of the positive and negative class accuracies.

After the iterations end, the final PP is then used over the test set to determine the reported values of the overall test accuracy a_{ts} and the test set g_{ts} value, that will measure how well balanced are the generalization abilities of the just constructed classifier. The pseudocode of the general procedure including noisy borderline patterns is thus:

```
trSetReduction(trainingSet tr, testSet ts)
  gTr = 0;
  acc+_Tr = 0;
  trainPP(ts, g, acc+, W, gamma);       // first update of weigths, margin
  while g >= g_Tr and acc+ >= acc+_Tr:  // reduce Tr while g, acc+ improve
    gTr = g; acc+_Tr = acc+; wPP = W;   // W_PP: weights of best PP so far
    find(Tr, R, N, B, nB-, gamma);      // find redundant, l. noisy, borderline
    remove(tr, R, N);                   // remove redundant, label noisy
    remove(tr, nB-);                    // and negative noisy boderline
    trainPP(tr, g, acc+, W, gamma);
  calcAccG(ts, wPP, accTs, acc+_Ts, acc-_Ts, gTs);
```

In the next section we will illustrate numerically these procedures.

4 Numerical results

We shall use 7 problem sets from the well known UCI database (listed in table 1) referring to the UCI database documentation [9] for more details on these problems. Some of them (glass, vowel, vehicle, thyroid) are multi–class problems; to reduce them to 2–class problems, we are taking as the minority classes the class 1 in the vehicle dataset, the class 0 in the vowel data set, and the class 7 in the glass domains (as done in [7]), and merged in a single class both sick thyroid

Problem set	initial g	final g	initial Tr set	final Tr set	ave. # iters
cancer	96.8	96.6	629	174	1.99
diabetes	69.9	71.2	691	631	0.88
ionosphere	76.9	82.3	315	241	2.13
vehicle	67.0	72.5	762	284	2.89
glass	90.4	91.6	193	163	0.59
vowel	89.3	93.3	891	418	1.62
thyroid	68.1	97.2	6480	64	4.81

Table 2. Comparison of initial and final g values. The table also shows the training set reduction achieved.

classes. In general they can be considered relatively hard problems. Moreover, some of these problems provide well known examples of highly imbalanced positive and negative patterns, that difficult classifier construction, as discriminants may tend to favor the (much) larger negative patterns over the less frequently positive ones. This is the case of the glass, vowel, thyroid and, to a lower extent, vehicle problems. In all of them we will take the minority class as the positive one.

PP training has been carried out as a batch procedure. In all examples we have used 3 perceptrons and parameters $\gamma = 0.05$ and $\eta = 10^{-2}$; for the thyroid dataset, we have taken $\eta = 10^{-3}$. As proposed in [1], the η rate does not change if the training error diminishes, but is decreased to 0.9η if it augments. Training epochs have been 250 in all cases; thus the training error evolution has not been taken into account to stop the training procedure. Anyway, it has an overall decreasing behavior. In all cases we have used 10–times 10–fold cross validation. That is, on each training stage, the overall data set has been randomly split in 10 subsets, 9 of which have been combined to obtain the initial training set, the size of which has been decreased on each training iteration as described above. To ensure an appropriate representation of positive pattern, stratified sampling has been used. The final PPs' behavior has been computed on the remaining, unchanged subset, that we keep for testing purposes.

Recall that we have discussed two handling options for training patterns in the set nB_i^-, either to keep or remove them. Table 1 gives the average of the final g values obtained over each test set. It also gives the proportion of positive patterns and the final g values given by a standard multilayer perceptron for comparison purposes. It can be seen that final g values arrived at removing patterns in nB^- are consistently better. Moreover, they favourably compare with MLP g values: although much simpler (and much faster to train), final PP g values are slightly better than MLP values in two cases, slightly worse in another two and essentially the same in the remaining three.

All other results will be given for training set selection when the nB_i^- sets are removed. They are contained in tables 2 and 3. The first table compares initial and final g values. In all cases but the cancer data set, final test g values are bigger than initial ones. The gain is small in some problems, that require

Problem set	initial acc	initial a^+	initial a^-	final acc	final a^+	final a^-
cancer	96.870	96.630	97.009	96.420	97.155	96.062
diabetes	75.039	57.926	84.369	75.158	61.373	82.665
ionosphere	82.143	64.051	92.367	86.171	71.803	94.255
vehicle	78.119	51.414	87.437	79.512	61.139	85.879
glass	95.048	84.500	96.795	95.842	86.000	97.459
vowel	97.273	80.667	98.933	97.838	88.111	98.811
thyroid	95.499	46.708	99.405	98.493	95.625	98.722

Table 3. Initial and final accuracy results for training set selection when patterns in nB^- are removed.

few training set selection iterations, but much larger in other cases; the average number of iterations is nevertheless quite modest. For a quick comparison, we just mention that the g values for the vehicle and vowel problems are better than those in [7], where a different training set reduction method is used with the 1–nearest neighbor (NN) and C4.5 algorithms; the glass g value reported here is slightly smaller than that reported there for the 1–NN method but better than that of C4.5 (notice that training sets used here may slightly differ from those used in [7]). On the other hand, except in the glass and diabetes problems training set reduction (shown in the same table) is quite marked, specially for the thyroid data set.

Table 3 compares initial and final accuracy values. In all cases final accuracy is bigger, except again for the cancer problem, where it remains essentially the same. As it should be expected, the algorithm enforced gain on the accuracy a^+ of the positive training class extends to the test sets, that show a noticeable increase, quite markedly in fact in all cases except the cancer dataset. On the other hand, the accuracy a^- of the negative class slightly increases in two cases, slightly decreases in another three and essentially stays the same in the remaining two cases.

As a summary of these results, we have illustrated that the proposed iterative training set selection procedure can achieve both noticeable improvements on the classification of a smaller positive class, while offering a good balance between positive and negative classification performances. Moreover, considerable reduction of training set sizes (and consequently a much faster training in the final iterations) are to be added to these advantages.

5 Conclusions and further work

In this paper we have proposed a new procedure for training set reduction based on the activation margins that arise naturally in parallel perceptron training. Its effectiveness has been verified on the seven 2–class problems studied here, several of them being representative of imbalanced class problems, where the discrimination of a small positive class may be damaged by the much larger number of negative samples. The proposed procedure balances in a natural way the number

of positive and negative samples while ensuring a good generalization, not only in terms of a good overall test set accuracy, but also of its adequate balance among positive and negative classes.

This property, together with the very fast training of PP, may make them quite useful on large dimension imbalanced problems, an area of considerable interest as many interesting problems (text mining, microarray discrimination) belong to it. This and other questions, such as PP use in active training, and improvements in their performance, either by combining PPs through boosting or enlarging their parameter space, are under study.

References

1. P. Auer, H. Burgsteiner, W. Maass, *Reducing Communication for Distributed Learning in Neural Networks*, Proceedings of ICANN'2002, Lecture Notes in Computer Science 2415 (2002), 123–128.
2. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, **Classification and Regression Trees**, Wadsworth, 1983.
3. N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, *SMOTE: Synthetic Minority Oversampling Technique*, Journal of Artificial Intelligence Research 16 (2002), 321–357.
4. J. Dorronsoro, F. Ginel, C. Sánchez, C. Santa Cruz, *Neural Fraud Detection in Credit Card Operations*, IEEE Transactions on Neural Networks, 8 (1997), 827-834.
5. T. Fawcett, F. Provost, *Adaptive Fraud Detection*, Journal of Data Mining and Knowledge Discovery 1 (1997), 291–316.
6. Y. Freund *Boosting a weak learning algorithm by majority*, Information and Computation 121 (1995), 256–285.
7. M. Kubat, S. Matwin, *Addressing the Curse of Imbalanced Training Sets: One-Sided Selection*, Proceedings of the 14th International Conference on Machine Learning, ICML'97 (pp. 179-186), Nashville, TN, U.S.A.
8. M.A. Maloof, *Learning when data sets are imbalanced and when costs are unequal and unknown,* ICML-2003 Workshop on Learning from Imbalanced Data Sets II, 2003.
9. P. Murphy, D. Aha, *UCI Repository of Machine Learning Databases*, Tech. Report, University of Califonia, Irvine, 1994.
10. N. Nilsson, **The Mathematical Foundations of Learning Machines**, Morgan Kaufmann, 1990.
11. J.A. Swets, *Measuring the accuracy of diagnostic systems*, Science 240 (1998), 1285–1293.
12. G.M. Weiss, F. Provost, *The effect of class distribution on classifier learning*, Technical Report ML-TR 43, Department of Computer Science, Rutgers University, 2001.