A Conservative Approach to Perceptron Learning

RAMASUBRAMANIAN G. SUNDARARAJAN[1] ASIM K. PAL[2]
[1] Information & Decision Technologies Lab
GE Global Research
Plot 122, EPIP Phase 2, Hoodi Village, Whitefield Road, Bangalore 560066
INDIA

[2] Professor of Information Systems & Computer Science
Indian Institute of Management Calcutta
D. H. Road, Joka, Kolkata 700104
INDIA

Abstract: - In many real-life pattern recognition problems, it may be prudent to reject an example rather than run the risk of a costly potential misclassification. Typically, the threshold for rejection is determined after the underlying classifier has been trained in order to minimize misclassification. In this paper, we present two algorithms to train a hyperplane with a bandwidth for rejection, wherein both the classifier and the bandwidth are determined simultaneously. Experimental results indicate that the hypothesis thus detrmined shows an improvement over its liberal counterpart (the same perceptron, with zero bandwidth), and over a perceptron trained using the standard perceptron learning rule, on which the rejection threshold is determined using Chow's rule.

Key-Words: - Perceptron learning, rejection threshold, emdedded reject option

1 Introduction

The primary focus of learning theory has been on cases wherein a learning algorithm outputs a prediction for every test example. However, this may not always be possible in real situations. In cases wherein a learning algorithm encounters regions of high ambiguity, or previously unseen regions of the input space, it may not have the confidence to make a prediction and may therefore prefer to reject the test example, i.e., not return a prediction. In certain situations, such as when the loss that might be incurred due to a wrong prediction is unacceptably high, conservatism may be a better policy. This approach is referred to in this paper as *conservative* learning.

Let $S = ((x_1, y_1), \ldots (x_\ell, y_\ell))$ be a labeled i.i.d. sample drawn from an unknown but fixed joint distribution $F(x, y),\ x \in X,\ y \in Y = \{1, \ldots m\}$, where m is the number of classes. A learning algorithm A uses the sample S to arrive at a hypothesis $h \in H$, which, in case a reject option is permitted, will output $h(x) \in \{0\} \cup Y$, where 0 represents the reject option. For pattern recognition problems, the simplest definition of L can be given as

$$L(x, y, h) = \begin{cases} 0 & \text{if } h(x) = y \\ \gamma & \text{if } h(x) = 0 \\ 1 & \text{if } h(x) \neq y, h(x) \neq 0 \end{cases} \quad (1)$$

where $0 < \gamma < 1$ represents the cost of rejection. A generalized version of this loss function can be written as:

*This work was done as part of the fellow (doctoral) program at IIM Calcutta

$$L(x, y, h) = \begin{cases} -a_y & \text{if } h(x) = y \\ c_y & \text{if } h(x) = 0 \\ b_y & \text{if } h(x) \neq y, h(x) \neq 0 \end{cases} \quad (2)$$

where $a_y > 0,\ 0 \leq c_y < b_y$.

The risk of the hypothesis h, expressed as the expectation of the loss L over the example space with respect to the probability measure F, is given by:

$$R(h) = \int L(x, y, h)\, dF(x, y) \quad (3)$$

The problem of learning with a reject option can now be restated as one of finding

$$h_{opt} = \arg \min_{h(h^p,\theta) \in H} R(h) \quad (4)$$

given a set of examples S. The basic inductive principle used to perform this task is Empirical Risk Minimization (ERM), which states that the best choice of hypothesis is the one that optimizes this risk for the given set of examples.

Intuitively, we understand that the applicability of the reject option depends on the confidence of the classifier in its prediction on a particular example. The appropriate measure of confidence depends on the problem, the nature of the underlying classifier, and the rationale behind the rejection scheme (ie., ambiguity, distance etc.). In order to represent this hierarchy, we consider a hypothesis with the reject option as being implemented using two hypotheses - the underlying classifier h^p, which returns a prediction on every example, and the rejection hypothesis θ, which returns a value of 1 is the example is to be rejected, and 0 otherwise.

$$h(x) = (1 - \theta(x, h^p))h^p(x) \quad (5)$$

A simple example of this is a perceptron with a bandwidth within which examples are rejected.

$$h(x, \alpha) = \begin{cases} +1 & \text{if} \quad \langle \mathbf{x}, \omega \rangle \geq \lambda_{+1} \\ -1 & \text{if} \quad \langle \mathbf{x}, \omega \rangle \leq -\lambda_{-1} \\ Reject & \text{otherwise} \end{cases} \quad (6)$$

where $\alpha = (\omega, \lambda), \lambda \geq 0$ reperesents the set of free parameters to be optimized, $\langle \cdot, \cdot \rangle$ denotes inner product and $\mathbf{x} = \phi(x)$ is a feature vector formed from the input vector. For example, a simple feature vector can be formed by taking the input vector x and augmenting it with an extra dimension always set to 1, so as to account for the intercept term in the perceptron equation. Alternatively, one may also represent h in terms of two parallel hyperplanes (characterized by two intercepts, with the rest of the weight vector remaining the same), rather than an intercept and two bandwidth terms. Such a representation would directly characterize h instead of through h^p and θ.

The rejection in case of h may be because the hypothesis is considered too simple to be accurate, but there isn't enough data to validate a more complex hypothesis. An alternative argument may also be that, even if a hyperplane is sufficient, the number of examples considered may be too small to position the hyperplane accurately enough. Therefore, even for a linearly separable sample, a good hypothesis may be one that allows for some non-zero bandwidth on either side of the hyperplane.

It is intuitive that solving the optimization problem (4) with respect to h^p and θ together would give a better solution than finding the underlying classifier h^p first, and then finding the appropriate θ for that classifier. In this paper, we present two methods to find both θ and h^p together for the case wherein h is a hyperplane with bandwidth as represented in equation (6).

2 Literature survey

Rejection of examples by a hypothesis is a fairly well-explored part of learning, and many strategies are often employed in practice to deal with such situations. Typically, the hypothesis is learnt from the examples without an embedded reject option, and an appropriate rejection threshold is set to operate on the strength of the output. (In other words, h^p and θ are learnt in sequence rather than simultaneously.) The classical work in this regard is by Chow, who analyzed the nature of the error-reject curve, and proposed that, if the a posteriori class probabilities are exactly known, the rejection threshold is given by

$$T = \frac{b - c}{b + a} \quad (7)$$

where a is the gain from a correct classification, b is the loss from a misclassification, and c is the loss from rejection [1]. However, this rule does not prove optimal in cases where there is significant overlap among the different classes, and the estimated a posteriori probabilities from the training data may not be accurate.

This risk is often mitigated by the use of voting schemes on multiple classifiers. Where a single classifier is concerned, the issue of having multiple reject thresholds for different classes has been explored; however, finding the optimal values for these thresholds can be a non-trivial problem [2]. Other strategies include using a differential cost function that reflects the conservatism that may arise because misclassifying a positive example may not have the same cost as misclassifying a negative example.

Recently, there has been work on training a Support Vector Machine (SVM) with an emdedded reject option, i.e., h^p and θ are trained simultaneously [3]. However, while SVMs are known to provide good generalization capabilities, they also take a much longer time to train. Therefore, perceptrons with bandwidth provide a middle ground between the efficiency of the simple perceptron algorithm and the accuracy of SVMs.

There exist algorithms in the literature that build on the classical perceptron algorithm by requiring the hyperplane to classify with a certain predefined bandwidth. More recent work has also focused on this problem, as evidenced by Krauth & Mezard's Perceptron Algorithm with Margins (PAM). This algorithm tries to find the hyperplane such that all examples are classified correctly with a bandwidth greater than a predefined quantity. Enhancements of this algorithm, such as the Perceptron Algorithm with Uneven Margins (PAUM) explores the possibility of allowing for asymmetric but fixed bandwidth on either side of the hyperplane [7]. The Approximate Large Margin Algorithm (ALMA) allows for symmetric but dynamically changing margins based on the misclassification count [4]. However, while the decision surface arrived at by these algorithms takes a form similar to the one presented in this paper, it does not take into account, the cost structure described in equations (1) or (2).

In the context of the literature cited above, the algorithm presented in this paper can be viewed as one of training a perceptron with an embedded reject option using class-related thresholds (implemented as asymmetric bandwidth).

3 The conservative perceptron learning rule (CPLR)

The algorithmic framework presented in this paper can be summarized in the following steps:

Step 1 $k = 0,\ \omega^k,\ \lambda^k$ random

Step 2 Repeat the following steps until convergence

2.1 $k = k + 1$
2.2 $\omega^{k+1} = \omega^k + \rho_\omega \Delta\omega^k$
2.3 $\lambda^{k+1} = \lambda^k + \rho_\lambda \Delta\lambda^k$

It remains for us to specify the rules by which $\Delta\omega^k$ and $\Delta\lambda^k$ are to be determined. In this paper, we present two algorithms that provide these rules.

3.1 CPLR 1

In this algorithm, both the weights and the bandwidths are learnt through gradient descent on the same criterion function. The criterion function is derived using the following rationale: If we were to look at the hypothesis h as a couple of parallel hyperplanes, then the distance of an example from the hypothesis would have to account for the distance from both hyperplanes. For a rejected/misclassified example, while one hyperplane provides the distance with respect to correct classification, the other hyperplane provides the distance with respect to misclassification/rejection. Therefore, the cost associated with example (x^k, y^k) can be written as:

$$J(x^k, y^k, \alpha^k) = \begin{cases} (a_{y^k} + c_{y^k})\tau_c^k - (b_{y^k} - c_{y^k})\tau_r^k \\ \text{if } h(x^k, \alpha^k) = 0 \\ (a_{y^k} + b_{y^k})\tau_c^k - (b_{y^k} - c_{y^k})\tau_r^k \\ \text{if } h(x^k, \alpha^k) \neq y^k \end{cases} \tag{8}$$

where

$$\tau_c^k = \lambda_{y^k}^k - y^k \langle \mathbf{x^k}, \omega^k \rangle$$

$$\tau_r^k = \lambda_{-y^k}^k + y^k \langle \mathbf{x^k}, \omega^k \rangle$$

The learning rules are arrived at by gradient descent on the criterion function described above.

$$\Delta\omega^k = \begin{cases} (a_{y^k} + b_{y^k})y^k\mathbf{x^k} & \text{if } h(x,\alpha) = 0 \\ (a_{y^k} + 2b_{y^k} - c_{y^k})y^k\mathbf{x^k} & \text{if } h(x,\alpha) \neq y \end{cases} \tag{9}$$

$$\Delta\lambda^k_{y^k} = \begin{cases} -(a_{y^k} + c_{y^k}) & \text{if } h(x,\alpha) = 0 \\ -(a_{y^k} + b_{y^k}) & \text{if } h(x,\alpha) \neq y \end{cases} \tag{10}$$

$$\Delta\lambda^k_{-y^k} = (b_{y^k} - c_{y^k}) \tag{11}$$

3.2 CPLR 2

In this algorithm, we learn the orientation of the hyperplanes (characterized by the weights and the intercept) on the basis of the distance between the example and hyperplane w.r.t. correct classification, weighted by the appropriate relative cost.

The criterion function to be minimized for weight learning is:

$$J(x^k, y^k, \alpha^k) = \begin{cases} (a_{y^k} + c_{y^k})\tau^k_c & \text{if } h(x^k,\alpha^k) = 0 \\ (a_y + b_y)\tau^k_c & \text{if } h(x^k,\alpha^k) \neq y^k \end{cases} \tag{12}$$

The learning rule for weights is given by gradient descent on the above criterion function, as follows:

$$\Delta\omega^k = \begin{cases} (a_{y^k} + c_{y^k})y^k\mathbf{x^k} & \text{if } h(x^k,\alpha^k) = 0 \\ (a_{y^k} + b_{y^k})y^k\mathbf{x^k} & \text{if } h(x^k,\alpha^k) \neq y^k \end{cases} \tag{13}$$

For learning the bandwidth, we examine the trade-off between correctly classified and misclassified examples on the predicted and rejected regions. Here we raise the question: what is the *relevant* set of examples needed for training? Since our interest is in the trade-off between rejection and classification (correct or wrong), the examples which are correctly classified and beyond the region of misclassified examples do not affect the trade-off in any way. The rationale for this statement is that the bandwidth can be increased to cover the most misclassified example, thereby achieving perfect classification on the predicted set, and any further increase in the bandwidth would not improve the solution. Therefore, on the predicted region, we constrain the learning rule to operate on examples whose net input $\langle \mathbf{x}, \omega \rangle$ is less than or equal to:

$$\xi^k_y(\omega^{k+1}) = \max_{sgn(\langle \mathbf{x},\omega^{k+1}\rangle)\neq y} |y\langle \mathbf{x}, \omega^{k+1}\rangle| \tag{14}$$

where this maximum is computed over the training sample. The bandwidth parameters are learnt in batch mode, on the constrained region of the example space as described by equation (14). The learning rule for the bandwidth parameters can be viewed in the following manner: On both the predicted and the rejected regions of the space, we must gain more than we lose, on the whole. Additionally, we impose the constraint that the predicted region can only make a positive bandwidth correction, whereas the rejected region can only make a negative bandwidth correction. The rationale for this constraint is that the bandwidth correction calculated on the predicted region does not take into account the trade-off in the rejected region, and vice-versa.

The method of calculation for the bandwidth correction on one side, $\Delta\lambda^k_{+1}$, is given here. The same method is to be followed on the other side as well.

Let the relevant predicted region $X^{k(p)}_{+1}$ be defined as:

$$X^{k(p)}_{+1}(\omega^{k+1}, \lambda^k_{+1}) = \{x | \lambda^k_{+1} \leq \langle \mathbf{x}, \omega^{k+1}\rangle \leq \xi^k_{-1}(\omega^{k+1})\} \tag{15}$$

Let p_1 and p_2 be the number of correctly classified examples and wrongly classified examples in $X^{k(p)}_{+1}$ respectively. The bandwidth correction for the predicted region on that side of the hyperplane is given by:

$$\Delta\lambda^{k(p)}_{+1} = \frac{1}{p_1 + p_2}(p_2(b_{-1} - c_{-1}) - p_1(a_{+1} + c_{+1})) \tag{16}$$

The bandwidth correction for the rejected region $\Delta\lambda^{k(r)}_{+1}$ is similar to that given in equation (16), except that the trade-off is in the opposite direction, and there is no constraint ξ^k_{-1} applicable on the rejected region. The eventual bandwidth correction is calculated according to the following equation:

$$\Delta\lambda^k_{+1} = max(\Delta\lambda^{k(p)}_{+1}, 0) + min(\Delta\lambda^{k(r)}_{+1}, 0) \quad (17)$$

The bandwidth correction rules described here have been given for the case of asymmetric bandwidth; the same rules can be applied to the case of symmetric bandwidth as well, with the only modification being that the correction due to the predicted and rejected regions are calculated using examples on both sides of the hyperplane together.

3.3 Convergence

One of the desirable properties of a training rule for a perceptron is that, on a linearly separable training sample, it should converge to a solution that perfectly classifies the sample. In order to achieve convergence for the two CPLR versions described above, we employ the following procedure while implementing the algorithm: After every weight learning step, we check if the liberal hyperplane has converged to a separable solution. If it has, then we freeze the weights and continue only with the bandwidth learning until all examples are correctly classified. For CPLR1, we also set $\Delta\lambda^k_{-y^k} = 0$ in order to ensure that there is only a decrease in bandwidth in each step.

By an extension of Novikoff's convergence proof for the perceptron algorithm [5], it can be proved that, if the bandwidth parameter is restricted to remain within upper bounds $(\bar{\lambda}_{-1}, \bar{\lambda}_{+1})$, both versions of the CPLR converge to a separating solution within a finite number of iterations.

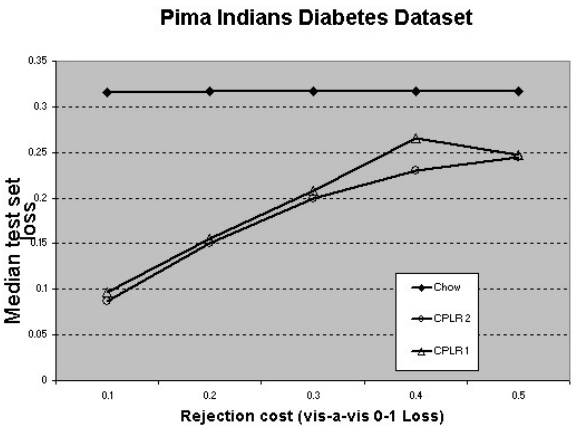
4 Experimental results

The two versions of the algorithm presented in Section 3 have been tried out on the Pima Indians Diabetes benchmark data set. This dataset was taken from the UC Irvine machine learning data repository. The prediction problem here is to determine whether or not a patient has diabetes, given a set of 8 numeric attributes. There were 768 examples in the data set, of which 568 were used for training and 200 for testing. In order to benchmark these algorithms, we compare them against a perceptron trained using the standard perceptron learning rule (PLR), upon which a threshold was applied using Chow's rule.

We randomly split the data into training and test sets 10 times, and run the CPLR and PLR on the training set. Since both algorithms perform a local descent on the criterion function, on each training-test split, we run the algorithm many times and choose the test set result corresponding to the best training set result obtained, in order to escape the problem of being trapped in bad local minima. Then we take the median result obtained across the 10 training-test splits, as representative of the performance of the algorithm.

For this experiment, we considered symmetric costs for both classes; therefore, the cost structure can be characterized simply by the rejection cost $0 < \gamma < 1$, vis-a-vis a $0-1$ loss function for the underlying classifier. The two algorithms were run for values of γ varying between 0.1 and 0.5. For $\gamma \geq 0.5$, application of Chow's threshold produces zero bandwidth; however, we also find that in most cases, the best solution of CPLR also produces a zero bandwidth hyperplane when $\gamma > 0.5$. For $\gamma \leq 0.5$, CPLR consistently outperforms its liberal counterpart.

The result of this experiment is depicted in the figure below. It can be seen that the performance of the perceptron algorithm with Chow's rule is fairly stable across values of γ, as is to be expected. However, the performance if CPLR seems to be related to the cost of rejection. The experimental results seem to indicate that CPLR performs better when there is more gain to be had from rejection.



4.1 Confidence Bounds

Here, we discuss some confidence bounds for comparing the performance of a conservative hypothesis (arrived at by CPLR or any other method) with its liberal counterpart. The method applied here is an extension of the one described by Vapnik & Bottou [8].

The hypothesis obtained by CPLR can be compared with its liberal counterpart on a validation set, and chosen if it exhibits an improvement in the net loss L. It is intuitive that if this improvement is observed, it would be due to the fact that the net loss of the liberal hypothesis on the rejected region is greater than the cost of rejection. While empirical evidence suggests that we accept the conservative hypothesis, we wish to be able to bound the probability that the actual error rate on the rejected region (see equation) does not exceed the rejection cost. We shall present here, the calculation for the case where the costs are symmetric. This can be extended to the asymmetric cost situation.

Let ℓ_1 be the number of rejected examples that are correctly classified by the liberal hypothesis, and ℓ_2 be the number of rejected examples that are misclassified by the liberal hypothesis. From the Chernoff-Hoeffding bounds [6], we get:

$$Pr\{\frac{\ell_2}{\ell_1+\ell_2} - \mu > \varepsilon\} < e^{-2\varepsilon^2(\ell_1+\ell_2)} \qquad (18)$$

where μ is the expected error rate on the rejected region. By setting $\varepsilon = \ell_2/(\ell_1+\ell_2) - \gamma$, we get the required confidence bound.

5 Scope for further work

The utility of a conservative approach to learning is determined by the type of problem being approached. This work describes a simple method for quantifying that utility and optimizing it locally using a linear separator. The algorithms described here have shown promising results on a benchmark data set, thereby suggesting the utility of further exploration along these lines.

Generalized versions, such as modeling the output as a real-valued variable, and the application thereof, of a smoothed version of the gain function cen be tried. This can then be used to extend the method to train multilayer perceptrons with an embedded reject option. Also, the SVM formulation with an embedded reject option described by Fumera & Roli can be generalized to account for asymmetric bandwidth and cost [3]. Optmizing the bandwidth with respect to the sample size can also be explored. Another area that can be studied is the trade-off between conservatism and complexity of the hypothesis class. A hypothesis may choose to be conservative if it finds that the problem is too complex to handle. The principle of conservatism can therefore be used to develop robust constructive algorithms, wherein the complexity is increased on local regions where the simpler hypothesis remains conservative.

References

[1] C. K. Chow, On optimum recognition error and reject trade-off, *IEEE Transactions on Information Theory*, vol. 16, pp. 41–46, 1970.

[2] G. Fumera and F. Roli, Multiple reject thresholds for improving classification reliability, Univ. of Calgary, Tech. Rep., 1999.

[3] G. Fumera and F. Roli, Support vector machines with embedded reject option, in *Pattern Recognition with Support Vector Machines - First International Workshop, Proceedings*, S.-W. Lee and A. Verri, Eds. Springer, 2002.

[4] C. Gentile, A new approximate maximal margin classification algorithm, *Journal of Machine Learning Research*, vol. 2, 2001.

[5] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. MIT Press, 1995.

[6] W. Hoeffding, Probability inequalities for sums of bounded random variables, *Journal of the American Statistical Association*, vol. 58, pp. 13–30, 1963.

[7] Y. Li, H. Zarazoga, R. Herbrich, J. Shawe-Taylor, and J. Kandola, The perceptron algorithm with uneven margins, NeuroCOLT, Tech. Rep., 2002.

[8] V. N. Vapnik and L. Bottou, Local learning algorithms, *Neural Computation*, vol. 4, no. 6, pp. 888–900, 1992.