ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2006, Vol.35, No.3A

A NOTE ON A LARGE MARGIN PERCEPTRON ALGORITHM

Bernd-Jürgen Falkowski

School of Economics, University of Applied Sciences Stralsund,
Zur Schwedenschanze 15, D-18435 Stralsund, Germany

Abstract. The importance of classification algorithms in the context of risk assessment is briefly explained. As an alternative to the popular support vector machines fault tolerant perceptron learning is suggested. In order to achieve better generalization properties the additional use of an iterative large margin perceptron algorithm is investigated. In particular it is shown that care has to be taken when initializing the algorithm. Some preliminary experimental results are briefly discussed.

1. Introduction

In the context of risk assessment for financial institutions, insurance agencies and the likes sophisticated classification algorithms have become increasingly important since large competitive advantages may be achieved, cf. e.g. [16], [18], [1], [10]. In the simplest case, the problem to be solved consists of classifying customers into bad, respectively, good risks according to data characterizing them. Usually these data are contained in vectors ("patterns") in a Euclidean space whose entries typically contain information concerning income, age, etc. Then, using suitable training data (i.e. vectors characterizing customers together with their a posteriori known risk classification), frequently a (generalized) linear discriminant is constructed which allows the computation of a weighted average that is then compared with a cut-off and thus a decision good/bad risk is arrived at according to whether the cut-off is exceeded or not.

The weights for this cut-off may be obtained by various methods: Classical statistical techniques (in particular Bayes theorem and logistic regression) may be employed as well as neural network techniques, cf. e.g. [6], [7], [2]. Recently so-called support vector machines (SVMs) have received much attention, cf. [5],[17], [11]. However, although these SVMs are large margin algorithms (in the sense that the separating hyperplane that constitutes the discriminant achieves a maximal separating margin, which promises good generalization capabilities, cf. [17]) they have some disadvantages. In particular, they are rather demanding from a programming point of view and their fault-tolerant version (the soft margin SVM) makes use of some approximations that are difficult to justify in general, cf. [17]. Hence as an alternative fault tolerant perceptron learning, cf. e.g. [8], [6], [7] has been investigated. Unfortunately,this algorithm does not possess the large margin property. Hence in this paper the additional use of an iterative large margin perceptron algorithm as proposed by Krauth and Mezard, cf. [13] is treated. The algorithm was originally devised in a physics context and needed some slight modifications since it was applied to rather specialized training data. Thus it became necessary to rework the complete proof using in particular a non-trivial initialization. Since during this process it turned out that a modified Krauth/Mezard algorithm as presented in [9] without proof contained a statement that could not be verified this did seem entirely justified though.

2. The Problem

Assume that suitable training data are available. In abstract terms, then p vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ from $\Re^n$ together with their risk classification ("good" respectively "bad" vectors associated to customers corresponding to class C_1, respectively, class C_2) are given, where no special restrictions on the vectors are assumed. Hence implicitly a preference relation (partial order) "$\rangle$" is determined for these vectors by

$$\mathbf{x}_i \rangle \mathbf{x}_j \quad \text{if} \quad \mathbf{x}_i \in C_1 \text{ and } \mathbf{x}_j \in C_2$$

It is then required to find a map $m_{\mathbf{w}}: \Re^n \to \Re$ that preserves this preference relation, where the index $\mathbf{w}$ of course denotes a weight vector. More precisely, one must have

$$\mathbf{x}_i \rangle \mathbf{x}_j \quad \Rightarrow m_{\mathbf{w}}(\mathbf{x}_i) > m_{\mathbf{w}}(\mathbf{x}_j)$$

If one now specializes by setting $m_{\mathbf{w}}(\mathbf{x}) := \langle \boldsymbol{\varphi}(\mathbf{x}), \mathbf{w} \rangle$, denoting the scalar product by $\langle .,. \rangle$ and an embedding of $\mathbf{x}$ in a generally higher dimensional feature space by $\boldsymbol{\varphi}$, then a linear discriminant is obtained. Here the weight vector $\mathbf{w}$ and a cut-off α (if

both exist) may be computed e.g. by applying the perceptron learning algorithm, cf. e.g. [14], such that

$\mathbf{x} \in C_1$ if $<\varphi(\mathbf{x}), \mathbf{w}> > \alpha$

$\mathbf{x} \in C_2$ if $<\varphi(\mathbf{x}), \mathbf{w}> < \alpha$.

Note here that this problem can be reduced by a standard trick as follows.

Embed the vectors in a space whose dimension is one higher than that of the feature space by defining $\varphi(\mathbf{x}) \rightarrow \mathbf{y} := [\varphi(\mathbf{x}), -1]$ respectively $[-\varphi(\mathbf{x}), 1]$ if $\mathbf{x} \in C_1$ respectively $\mathbf{x} \in C_2$ and map the weight vector $\mathbf{w}$ to $\mathbf{w}_1 := [\mathbf{w}, \alpha]$. Then clearly the problem is reduced to finding a weight vector $\mathbf{w}_1$ such that $<\mathbf{y}, \mathbf{w}_1> > 0$ in the higher dimensional space. In the sequel, it is this reduced problem that will be considered.

Of course, it must be admitted that the existence of a suitable weight vector is by no means guaranteed. However, at least in theory, the map φ may be chosen such that the capacity of the perceptron is large enough for a solution to exist with high probability.

In order to see this, note that the number of different monomials of degree i in $x_1, x_2, \ldots, x_n$ is given by B(n+i-1,i), where B denotes the binomial coefficient, cf. e.g. [12], p. 488. Hence an easy induction proof shows that the number of different monomials of degree less than or equal to i in $x_1, x_2, \ldots, x_n$ is given by B(n+i,i). Thus φ may be defined by

$$\varphi(\mathbf{x}) = (1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_n^i)$$

as a map from $\Re^n$ to $\Re^{B(n+i,i)}$ and hence the separating capacity of the corresponding hyperplanes may be increased with i, for details see e.g. [4].

The price one has to pay for this increased separating capacity consists of larger computation times and, perhaps more importantly, loss of generalization capabilities due to a higher VC-dimension of the separating hyperplanes, cf. e.g. [17], [15]. Hence the importance of this procedure is of a more theoretical nature.

Indeed, in practical situations fault tolerant learning such as soft margin SVMs or Gallant's pocket algorithm will be preferred. For reasons given in the introduction here the latter will be further investigated. In particular, it will be of interest how, given the correctly classified patterns, large margin separation can be achieved. To this end, it was intended to employ a slightly modified version of the Krauth/Mezard algorithm initializing it with the discriminant weights obtained by Gallant's algorithm, since this seemed a good approximation to the optimal weights. This appeared immediately feasible, since in [9] it was claimed without proof that an arbitrary initialization is possible. However, a close investigation of the original paper of Krauth and Mezard where an initialization with the zero vector was used, revealed that in case of an arbitrary initialization there is a gap in the proof that at least the present author could not close. Hence a new proof (following the original one quite closely in most places but removing the restriction on the vectors) is presented below.

3. Generalization of the Krauth/Mezard Algorithm

Given samples (patterns) $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$, find a vector $\mathbf{w}^*$ such that

$< \mathbf{w}^*, \mathbf{x}_j > \geq c$ forj = 1, 2, …, p and a fixed constant $c > 0$.

Assume throughout that such a vector exists with $\| \mathbf{w}^* \| = c/\Delta_{opt}$, where Δ_{opt} denotes the maximal distance of separation of the samples from zero.

3.1. Algorithm

Set $x := \max_j \|\mathbf{x}_j\|^2$.

At t = 0 set $\mathbf{w}_0 = \lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 + \ldots + \lambda_p\mathbf{x}_p$ where all the λs are arbitrary but nonnegative.

At t = 0, 1, ... determine a pattern $\mathbf{x}_{j(t)}$ by

$< \mathbf{w}_t, \mathbf{x}_{j(t)} > := \min_j < \mathbf{w}_t, \mathbf{x}_j >$

and if

$< \mathbf{w}_t, \mathbf{x}_{j(t)} > < c$ (c is a fixed positive number)

use it to update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} = \mathbf{w}_t + (1/x)\mathbf{x}_{j(t)} \qquad (1)$$

If $< \mathbf{w}_M, \mathbf{x}_{j(M)} > \geq c$ stop.

The properties of the algorithm are a consequence of the following considerations, the first part of which is concerned with general convergence properties whilst the second part shows that a maximal margin of separation may be achieved by letting c tend to infinity. The arguments used in the first part are essentially well-known from perceptron learning whilst the method utilized in the second part is somewhat more complicated. In either case, clearly the Krauth/Mezard paper provided essential guidance.

3.2. General Convergence Properties

After M updates according to the rule (1) assuming that pattern $\mathbf{x}_j$ has been used m_j times where $\Sigma_j m_j = M$ one has the following

$$c^*(M+\Lambda)/x \leq (1/x)\,\Sigma_j (m_j + \lambda_j)<\mathbf{w}^*, \mathbf{x}_j> = <\mathbf{w}^*, \mathbf{w}_M> \leq (c/\Delta_{opt})^*\|\mathbf{w}_M\| \qquad (2)$$

where $\Lambda := \Sigma_j \lambda_j$.

On the other hand, an upper bound on $\|\mathbf{w}_M\|$ is given by considering

$$\|\mathbf{w}_{t+1}\|^2 - \|\mathbf{w}_t\|^2 = (2/x) < \mathbf{w}_t, \mathbf{x}_{j(t)} > + (1/x^2)\| \mathbf{x}_{j(t)}\|^2 \leq (2/x)^*c + 1/x = (2c + 1)/x$$

which implies

$$\|\mathbf{w}_M\|^2 \leq M^* (2c + 1)/x + \|\mathbf{w}_0\|^2 \qquad (3)$$

Now (2) and (3) together give an upper bound on M, namely

$$M \leq (x^2/\Delta_{opt}^2)^*\{ (2c + 1)/x + \|\mathbf{w}_0\|^2) \qquad (4)$$

Hence the algorithm converges in a finite number of steps (note that the upper bound given is not tight!).

Indeed, even more can be said by considering the margin of separation Δ_c determined by this algorithm:

$$\Delta_c \geq c/\|\mathbf{w}_M\| \geq [\Delta_{opt} * c * (M+\Lambda)/x] / \|\mathbf{w}_M\|^2 = \Delta_{opt} / A \quad (5)$$

Finally the A, which clearly is a performance guarantee factor, can be bounded using (3) and(4) above by

$$A = \|\mathbf{w}_M\|^2 / [c*(M+\Lambda)/x] \leq [M^* (2c + 1)/x + \|\mathbf{w}_0\|^2]/(c*M/x) \Rightarrow$$

$$A \leq (x/c)*[(2c + 1)/x + \|\mathbf{w}_0\|^2/M] \quad (6)$$

Note here that (6) implies that for large M the bound for A given in [13] namely $A \leq (2+1/c)$ is recovered.

3.3. Optimal margin of separation

In order to prove that for $c \to \infty$ Δ_c approaches Δ_{opt} following [13] $\mathbf{w}_t$ is decomposed as

$$\mathbf{w}_t = a(t)\mathbf{w}^* + \mathbf{k}_t \quad \text{where } <\mathbf{k}_t, \mathbf{w}^*> = 0. \quad (7)$$

The reasoning is analogous to the one employed above but one reasons separately for $\mathbf{w}^*$ and $\mathbf{k}_t$ by decomposing the term $(1/x)\mathbf{x}_{j(t)}$ accordingly.

First note that

$$a(t) = < \mathbf{w}_t, \mathbf{w}^* >/\|\mathbf{w}^*\| \quad (8)$$

and hence that $a(t) > 0$ for $t > 1$ and $a(0) \geq 0$ since $\mathbf{w}_t$ is nonnegative linear combination of patterns $\mathbf{x}_j$.

Next observe that $\mathbf{x}_{j(t)}$ always has a negative projection on $\mathbf{k}_t$ i.e.

$$< \mathbf{k}_t, \mathbf{x}_{j(t)} > < 0. \quad (9)$$

This is so since the condition $\min_u [<\mathbf{w}^* + u*\mathbf{k}_t, \mathbf{x}_{j(t)} >/\|\mathbf{w}^* + u*\mathbf{k}_t\|] < \Delta_{opt}$ for all $u \geq 0$ would be violated. This in turn can be seen since $f(u) := \min_u [<\mathbf{w}^* + u*\mathbf{k}_t, \mathbf{x}_{j(t)} >/\|\mathbf{w}^* + u*\mathbf{k}_t\|]$ satisfies this inequality by exploiting the positivity of a(t) and taking $u = 1/a(t)$. However, if $< \mathbf{k}_t, \mathbf{x}_{j(t)} > > 0$ would hold, then a simple differentiation shows that the minimum for f(u) is given for u = 0 and this is greater than or equal to Δ_{opt}.

Now (9) can be exploited by arguing on $\mathbf{k}_t$ as for (3) to get

$$\|\mathbf{k}_t\| \leq [t/x + \|\mathbf{k}_0\|^2]^{1/2} \quad (10)$$

If learning stops after M steps as assumed above, then a(M-1) can be bounded as follows

$$< \mathbf{w}_{M-1}, \mathbf{x}_{j(M-1)} > = a(M-1)< \mathbf{w}^*, \mathbf{x}_{j(M-1)} > + < \mathbf{k}_{M-1}, \mathbf{x}_{j(M-1)} > < c \Rightarrow$$

$$a(M-1) < [c + |< \mathbf{k}_{M-1}, \mathbf{x}_{j(M-1)}>|]/c \Rightarrow$$

$$a(M-1) \leq \{c + [(M-1) + x^*\|\mathbf{k}_0\|^2]^{1/2}\}/c \quad (11)$$

A bound on a(M) is obtained from the learning rule in the algorithm using (8):

$$a(M) - a(M-1) = < \mathbf{w}_M - \mathbf{w}_{M-1}, \mathbf{w}^* >/\|\mathbf{w}^*\|^2 = (1/x) |< \mathbf{x}_{j(M-1)}, \mathbf{w}^*>| /\|\mathbf{w}^*\|^2$$
$$\leq (x^{-1/2})*(1/\|\mathbf{w}^*\|) = \Delta_{opt}/(c*x^{1/2})$$

Thus using (11) one obtains

$$a(M) \leq \{c + [(M-1) + x^*\|\mathbf{k}_0\|^2]^{1/2}\}/c + \Delta_{opt}/(c*x^{1/2}) \quad (12)$$

and exploiting (11) and (12) that $\|\mathbf{w}_M\|$ is bounded by

$$\|\mathbf{w}_M\| \leq a(M)* \|\mathbf{w}^*\| + \|\mathbf{k}_M\| \Rightarrow$$

$$\|\mathbf{w}_M\| \leq \|\mathbf{w}^*\|*\{[c + [(M-1) + x^*\|\mathbf{k}_0\|^2]^{1/2}]/c + \Delta_{opt}/(c*x^{1/2}) + (\Delta_{opt}/c)*[M/x + \|\mathbf{k}_0\|^2]^{1/2}\}. \quad (13)$$

Finally it follows from (13) that

$$c/\Delta_{opt} \leq c/\Delta_c \leq \|\mathbf{w}_M\| \to \|\mathbf{w}^*\| = c/\Delta_{opt} \text{ as } c \to \infty \quad (14)$$

since M grows at most linearly with c, see (4).

4. Conclusion

Note that the nonnegativity of a(t), cf. (8), seems to be essential for the second part of the correctness proof, whilst for the first part (using mainly classical perceptron convergence arguments) this is not necessary. Hence it seems that after fault tolerant learning the weight vector obtained by Gallant's pocket algorithm cannot be exploited to initialize the Krauth/Mezard algorithm since it will contain samples that are not correctly classified in general. On the other hand, seeing that a suitable initialization might shorten computing times (this is suggested, for example, by the slick perceptron convergence proof given in [3]) it might be helpful to compute a solution of the inequality system by the usual perceptron learning first and then use this weight vector as initialization. No problems would occur in this case since the coefficients of the weight vector would satisfy the positivity condition. It seems that extensive experiments are required to resolve the issue and it is intended to conduct these as soon as suitably large realistic samples become available. At any rate, preliminary tests with approximately 360 "real life data" provided by a German financial institution showed that chosing c large enough (approximately 50) can increase the margin of separation quite significantly (> 50%) when compared to standard perceptron learning whilst using still negligible CPU times (1 to 2 minutes) when initializing with the zero vector. Incidentally, in [19] a method was suggested that was claimed to convert the pocket algorithm (or rather its dual) into a large margin algorithm. However, again the proof seems to contain a gap: In certain cases (which are not all that special) fault tolerant learning is still finite state and thus convergence cannot be guaranteed. Details concerning this statement will be reported elsewhere.

Acknowledgement: The presentation in the paper was improved following up the hints of three anonymous referees.

References

[1] Banking Committee on Banking Supervision: International Convergence of Capital Measurements and Capital Standards, A Revised Framework, Bank for International Settlements, *http://www.bis.org/publ/bcbsca.htm (June 2004).*

[2] **C.M. Bishop.** Neural Networks for Pattern Recognition. *Oxford University Press*, 1998.

[3] **H.D. Block, S.A. Levin.** On the Boundedness of an Iterative Procedure for Solving a System of Linear Inequalities. *Journal of the American Mathematical Society*, 26, 1970.

[4] **T.M. Cover.** Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Trans. on Electronic Computers, Vol*.14, 1965.

[5] **N. Cristianini, J. Shawe-Taylor.** An Introduction to Support Vector Machines and other Kernel-Based Learning Methods. *Cambridge University Press*, 2000.

[6] **B.-J. Falkowski.** Assessing Credit Risk Using a Cost Function. *Proceedings of the Intl. Conference on Fuzzy Information Processing, Beijing, Vol*.II, *Tsinghua University Press, Springer-Verlag*, 2003.

[7] **B.-J. Falkowski.** Scoring Systems, Classifiers, Default Probabilities, and Kernel Methods. *Proceedings of the* 2004 *Intl. Conference on Machine Learning and Applications* (ICMLA'04), *Eds. M. Kantardzic, O. Nasraoui, M. Milanova, IEEE Catalog Number*: 04EX970, 2004.

[8] **S.I. Gallant.** Perceptron-based Learning Algorithms. *IEEE Transactions on Neural Networks, Vol*. I, *No*. 2, 1990.

[9] **I. Guyon, D.G. Stork.** Linear Discriminant and Support Vector Classifiers. *Advances in Large Margin Classifiers (Eds. Smola, A.J.; Bartlett, P.L.; Schölkopf, B.; Schuurmanns, D.), MIT Press*, 2000.

[10] **D.J. Hand, W.E. Henley.** Statistical Classification Methods in Consumer Credit Scoring: a Review. *J.R. Statist. Soc. A*, 160, *Part* 3, 1997.

[11] **S. Haykin.** Neural Networks. second edition. *Prentice Hall*, 1999.

[12] **D.E. Knuth.** The Art of Computer Programming. *Vol*. 1, *Fundamental Algorithms*, 2*nd Edition*, 1973.

[13] **W. Krauth, M. Mezard.** Learning Algorithms with Optimal Stability in Neural Networks. *Journal of Physics A: Math. Gen*. 20, 1987.

[14] M.L. Minsky, S. Papert. Perceptrons. *MIT Press*, (*Expanded Edition*), 1990.

[15] **B. Schölkopf, A.J. Smola.** A Short Introduction to Learning with Kernels. *Mendelson, S.; Smola, A.J. (Eds.): Advanced Lectures on Machine Learning, Springer-Verlag, LNAI* 2600, 2003.

[16] **L.C. Thomas.**A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, 16, 2000.

[17] **V.N. Vapnik.** Statistical Learning Theory. *John Wiley & Sons*, 1998.

[18] **J. Voit.** The Statistical Mechanics of Financial Markets. 3*rd edition, Springer-Verlag*, 2005.

[19] **J. Xu, X. Zhang, Y. Li.** Large Margin Kernel Pocket Algorithm. *Proceedings of the International Joint Conference on Neural Networks* 2001, IJCNN'01, *Vol*. 2. *New York: IEEE*, 2001.

Received August 2006.