Perceptron Learning versus Support Vector Machines

Bernd-Jürgen Falkowski and Silvio Clausen
University of Applied Sciences Stralsund
Zur Schwedenschanze 15
D-18435 Stralsund
Germany
Email: Bernd.Falkowski@fh-stralsund.de and Silvio.Clausen@fh-stralsund.de

Abstract In this paper the performance of perceptron learning is compared with that of support vector machines if the pertinent Vapnik-Chervonenkis (VC-) Bound is small, thus complementing some known results. To this end the creditworthiness of bank customers is evaluated using real-life (anonymous) data and a so-called scoring system. Since in practical situations there is usually a shortage of available learning data as well as a lack of sophisticated software a comparatively small number of data is used for fault tolerant perceptron learning implemented in Excel using VBA. In order to justify the tests it is shown that the VC-Bound of the raw data is indeed small. The significance of experimental results with respect to applying perceptron learning as opposed to SVMs is discussed since it allows the use of cost functions. This is of particular relevance within the banking context.

Key words: support vector machines, perceptron learning, scoring systems

1 Introduction

Ever since the Basel II central banks agreement of the G10-states, cf. e.g. [4], the individual objective rating of the creditworthiness of customers has become an important problem To this end so-called scoring systems, cf. e.g. [8], have been used for quite some time. Generally these sytems are implemented as linear discriminants where customer characteristics such as income, property assets, liabilities and the likes are assigned points or grades and then a weighted average is computed, where a customer is judged "good" or "bad" according to whether the average exceeds a cut-off point or not. In an extreme case the attributes are just binary ones where 0 respectively 1 signifies that the property does not hold respectively holds. This situation frequently arises in pratice and will also apply here. The all important weights can then either be computed using classical statistical methods or more recently with the help of artificial neural networks provided that suitable bank records are available for "learning".
Unfortunately though in practice often there are only comparatively few records vailable and moreover, if one wishes to equip a large number of banks with suitable systems, there will be a shortage of sophisticated and hence expensive software. Thus it seems preferable to utilize a fault tolerant perceptron learning algorithm as described in [7] and implemented in [6] as opposed to the more sophisticated support vector machines. In addition this has the advantage of allowing the use of a cost function, cf. [6], which is very important for profit maximization in a banking context. However, the question remaining was of course, whether by adopting this procedure the generalization capabilities of the system would not be too severely restricted. Hence the experiments described below were conducted using comparatively few "real-life data", having established first that this could be justified by showing the VC bound to be small (Incidentally, one may well ask why it should have been desirable to keep the VC-Bound so small by using a linear discriminant. The reason is simply that it was thought necessary to keep the system transparent to the banking experts in order to ensure acceptance). The corresponding SVM was thus implemented with a trivial kernel, since a more sophisticated embedding in a higher dimensional space would have obscured the significance of the weights. This procedure made possible the use of the Excel Solver to deal with the quadratic programming problem. The results obtained complement some earlier ones given in [11] where a much larger VC-Bound applied. In addition they allow an interesting comparison between a probabilistic algorithm (fault tolerant perceptron learning) and a deterministic one (SVM)

2 The Raw Data and their Associated VC-Dimension

The raw learning data (anonymous) available from a large german bank were given as an Excel table whose first column contained a number to identify the individual data set whilst the other columns contained ones and zeros only. Thus there were only binary attributes with the last column indicating the rating of the customer based on past experience (0 for a "bad" customer, 1 for a "good" customer). These attributes arise in a natural way if one partitions "compound characteristics" as for example the age into several intervals indicating that the age belongs respectively does not belong to a certain interval by a one respectively zero as the binary attribute value.

The abstract set-up may then be described as follows: Suppose that the compound characteristics are denoted by $c_1, c_2, ..., c_n$. Further assume that the binary attribute values associated with compound characteristic c_i are c_{ij} and that the corresponding points scores (weights) to be computed by perceptron learning respectively a SVM are p_{ij}. If the (also to be computed) cut-off is t, then the decision procedure is described by:

A customer will be classified as "good" if

$$\Sigma_{i,j}\ c_{ij}*p_{ij} > t \qquad \text{---------------------}(*)$$

Thus one sees that the VC-Dimension of the set of separating hyperplanes is bounded above by (1 + number of binary attributes), cf. e.g. [5].

However, the decision procedure (*) given above may be converted into an equivalent one as follows.

Suppose that the domain of the p_{ij} is described by

$$\min_i \leq p_{ij} \leq \max_i \qquad \text{for all } i,$$

and introduce the following abbreviations:

$$\min_{av} := (1/n)*\Sigma_i \min_i,$$
$$\max_{av} := (1/n)*\Sigma_i \max_i,$$
$$s := \max_{av} - \min_{av}.$$

Then affine transformations t_{ij} of the points scores p_{ij} may be defined by:

$$q_{ij} = t_{ij}(p_{ij}) :=$$
$$\min_{av} + [(p_{ij} - \min_i)/(\max_i - \min_i)]*s.$$

Hence the following holds:

$$q_{ij} \in [\min_{av}, \max_{av}], \quad \text{and}$$
$$q_{ij} = \beta_i * p_{ij} + \alpha_i \quad \text{where}$$
$$\beta_i := s/(\max_i - \min_i),$$
$$\alpha_i := [1/(\max_i - \min_i)]*(\max_i*\min_{av} - \min_i*\max_{av})$$

Since t_{ij} is invertible with

$$p_{ij} = (1/\beta_i)*q_{ij} - \alpha_i/\beta_i$$

it follows that

$$\Sigma_{i,j}\ c_{ij}*p_{ij} > t \quad \Leftrightarrow$$
$$\Sigma_{i,j}\ c_{ij}*[(1/\beta_i)*q_{ij} - \alpha_i/\beta_i] > t$$

But if

$\Sigma_j c_{ij} = 1$ holds for all i, one has

$$\Sigma_{i,j}\ c_{ij}*[\alpha_i/\beta_i] = \Sigma_i [\alpha_i/\beta_i] = 0$$

and hence

$$\Sigma_{i,j}\ c_{ij}*[(1/\beta_i)*q_{ij} - \alpha_i/\beta_i] > t \quad \Leftrightarrow$$
$$\Sigma_{i,j}\ w_i*(c_{ij}*q_{ij}) > t \qquad \text{--------------------}(**)$$

where $w_i := 1/\beta_i$ is independent of j.

Note here that the condition $\Sigma_j c_{ij} = 1$ holds, of course, for all i, since the binary attributes had been arrived at by partitioning the domain of the compound characteristics.

If one now embeds the vectors of binary attribute values into $\Re^n$ by setting the corresponding entries to

$$x_i := \Sigma_j\ c_{ij}*q_{ij}$$

then, because the decision procedure (*) is equivalent to the procedure (**), the following holds:

2.1 Lemma:

If there are n compound attributes giving rise to the binary attributes as described above then the VC-Dimension of the set of separating hyperplanes is bounded above by n+1.

As a consequence of this one immediately obtains from the corollary to theorem 10.3 in [11] p. 408 the following:

2.2 Lemma

If l is the number of training examples that are correctly separated by the decision procedure (*) then with probability $1-\eta$ one can assert that the error rate has the upper bound

$$(2/l)*\{(n+1)*[\ln(2*l/(n+1))+1] - \ln(\eta/4)\}.$$

Remark: For the experiments conducted (see below) the reduction in the bound of the VC-Dimension on the set of separating hyperplanes possibly resulting from choosing large margins of separation and from the geometry of the binary vectors considered, cf. theorem 10.3 in [11] p. 408, could not be observed. Hence the above transformations to show a (comparatively) low VC-Bound seem all the more relevant.

3 Experiments

The experiments were carried out using 200 data sets of the format described in section 2 for learning and 200 similar data sets (not seen) for testing. There were 160 "good" and 40 "bad" customers in either data set.

The number of binary attributes used was 44 whilst there were only 14 compound attributes.

Assuming that perfect learning is possible for these attributes (as turned out to be the case) it follows from Lemma 2.2 that an upper bound on the error rate for the unseen data is given by ≈ 0,68 with a probability of 0.9. Of course, this is rather too large to give meaningful results. Bearing in mind, however, that this bound is thought to be far from tight in general, since it is distribution-free, cf. e.g. [10], p. 45, experiments were nevertheless carried out, since the nunber of examples required for a theoretical justification could not be obtained.

The algorithms compared were on the one hand a version of the pocket algorithm (fault tolerant perceptron learning algorithm), cf. e.g. [6] for some pseudo code, and on the other hand the algorithm implemented within the framework of the Excel Solver applied to solve the quadratic programming given below, thus constructing a SVM with a trivial kernel.

The main differences between these algorithms may be summarized as follows:

Fault tolerant perceptron learning uses a probabilistic algorithm whilst the Excel Solver employs a classical Newton algorithm. The former's complexity is not known although in all practical applications it has performed very well. The latter suffers from the Excel implementation since there are uncontrollable side effects created for example by cell updates. In both cases, however, CPU times turned out to be negligible

(in the order of a few minutes at most) in view of the small number of data.
The perceptron learning constructs a separating hyperplane if possible (minimizing the number of mistakes made, if a separating hyperplane does not exist) by solving a system of linear inequalities, cf. also [1].
The quadratic programming problem solved by the second algorithm is given as follows:
Maximize

$$Q(\alpha) := \Sigma_i \alpha_i - \Sigma_{ij} \alpha_i \alpha_j d_i d_j <\mathbf{x}_i, \mathbf{x}_j>$$

subject to

$$(1)\ \Sigma_i \alpha_i d_i = 0$$

and $(2)\ 0 \leq \alpha_i \leq C$

where <.,.> denotes the scalar product, the $\mathbf{x}_i$ are the Boolean vectors describing the bank customers, the d_i are +1 or –1 according to whether a “good” or a “bad” customer is being considered, and C is a constant whose value has to be experimentally optimized, for further details see e.g. [9], p. 324. If C is infinity then the solution of the quadratic programming problem allows one to construct a separating hyperplane fron the α_is as well. The difference to perceptron learning being that this hyperplane is the unique optimal hyperplane implementing a maximal margin of separation. Theoretical results show that the use of this hyperplane will generally lead to better generalization properties, cf. e.g. [2], [11]. The question to be (at least provisionally) anserwed by the experiments was: How large is the differnce between the two algorithms if the VC-Bound is rather small?

4 Results

The experiments were conducted on a standard Pentium IV PC with 1.4 GHz and 1 GB RAM.
As mentioned above the CPU times used were within the region of a few minutes although the fault tolerant learning turned out (perhaps slightly surprisingly) faster in all cases. Due to more or less unpredictable side effects like cell updates there seemed to be little point in comparing exact times.
The quadratic programming problem was tackled first with various C values and it was only discovered after lengthy computation that a perfect solution was possible (even C = 30 still leads to 0.5 % errors if the corresponding weight vector and cut-off is used to implement the decision procedure (*) above.
In contrast perceptron learning found a perfect solution if the number of iterations of the main loop of the algorithm was set to 50 000.
Both error rates above refer to the learning data. For the unseen data the error rate was of course significant as was to be expected from the considerations using the VC-Bound above.
In fact we obtained an error rate of 20.5% on the unseen Data using the SVM and an error rate of 22% using perceptron learning. Whilst this may seem disappointing at first sight, a comparison with the results given in [3] shows that error rates of this order of magnitude are by no means unreasonable within the credit risk assessment context considered.

5 Discussion

Consideration of the VC-Bound and in particular 2.2 above shows that the results obtained have to be treated with a certain amount of caution. Nevertheless in the light of other experimental results availble, cf. [3], they do seem at least of practical significance. This is all the more so, since there are, for obvious reasons, few results concerning the kind of banking application considered, publicly availble.
It is interesting to note that the improvement achieved concerning the generalization ability of the system whilst using an SVM seems to be comparatively small in the presence of a small VC-Bound as shown by the experimental results. Indeed, any differences observed during the experiments described above are most probably purely accidental.
Thus one might conclude that the disadvantages resulting from the construction of a possibly suboptimal separating hyperplane are outweighed by the advantages. The most noteworthy of these in the context considered are

(i) ease of implementation
(ii) few software requirements
(iii) possibility to use a cost function

Incidentally: The experiments showed that the experimental optimization of the constant C is somewhat unfortunate since one cannot know in advance if an optimal solution to the quadratic programming problem is possible.
It is hoped to conduct further experiments if additional data become available in order to verify the results obtained within a more reliable setting. In particular it would be interesting to consider the situation where perfect separation is no longer possible. The authors suspect that then SVMs may have even less of an advantage in a situation where the VC-Bound is already small due to the influence of the geometry of the data.
Of course, one would expect the situation to be somewhat different in view of the experimental results given in [11], if higher order sepration surfaces (or equivalently non-trivial kernels) could be utilized. However, as pointed out before, in a banking context it is rather difficult to gain acceptance for a system that it is not as transparent as possible. These objections arising entirely from psychological reasons must not be neglected. In addition it should be pointed out that even using linear systems combined with a cost function

quite impressive results can be achieved, for further details the reader may consult e.g. [6].

6 References

[1] Block, H.D.; Levin, S.A.: On the Boundedness of an Iterative Procedure for Solving a System of Linear Inequalities. Proc. of the AMS, 26, (1970)

[2] Christianini, N.; Shawe-Taylor, J.: An Introduction to Support Vector Machines and other Kernel-Based Learning Methods. Cambridge University Press, (2000)

[3] Episcopos, A.; Pericli, A.; Hu, J.: Commercial Mortgage Default: A Comparison of the Logistic Model with Artificial Neural Networks. Proceedings of the Third Intl. Conference on Neural Networks in the Capital Markets, London, England, (1995)

[4] Europäische Zentralbank: Die neue Basler Eigenkapitalvereinbarung aus Sicht der EZB. Monatsbericht, Mai, (2001)

[5] Falkowski, B.-J.: On Scoring Systems with Binary Input Variables.In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, Vol. XIII, International Institute of Informatics and Systemics, (2002)

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

[7] Gallant, S.I.: Perceptron-based Learning Algorithms. IEEE Transactions on Neural Networks, Vol. I, No. 2, (1990)

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

[9] Haykin, S.: Neural Networks,.2nd edition, Prentice Hall, (1999)

[10] Schölkopf,B.; Burges, C.J.C.; Smola, A.J.: Advances in Kernel Methods. MIT Press, (1999)

[11] Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, (1998)