Similarity Based Classification

Axel E. Bernal[1], Karen Hospevian[2], Tayfun Karadeniz[3], and Jean-Louis Lassez[3]

[1] University of Pennsylvania. CIS Department
Phildelphia, PA 19104. USA
axel@snowball.pcbi.upenn.edu

[2] New Mexico Institute of Mining and Technology. Computer Science Dept.
Socorro, New Mexico. USA
karen@nmt.edu

[3] Coastal Carolina University. Department of Computer Science
Conway, SC 29528. USA
{jlassez, tkarade}@coastal.edu

Abstract. We describe general conditions for data classification which can serve as a unifying framework in the study of kernel based Machine Learning Algorithms. From these conditions we derive a new algorithm called SBC (for Similarity Based Classification), which has attractive theoretical properties regarding underfitting, overfitting, power of generalization, computational complexity and robustness. Compared to classical algorithms, such as Parzen windows and non-linear Perceptrons, SBC can be seen as an optimized version of them. Finally it is a conceptually simpler and a more efficient alternative to Support Vector Machines for an arbitrary number of classes. Its practical significance is illustrated through a number of benchmark classification problems.

1 Introduction

Research in Machine Learning has recently received considerable attention due to massive proliferation of data in Bioinformatics and the Internet. In particular, Support Vector Machines (SVM) have been shown to play a major role in microarray data analysis [1], and web data classification [7]. We refer the reader to [5] for a thorough treatment and further references regarding the wide range of applications of SVM. The main significance of Support Vector Machines is their theoretical underpinning and their handling of non-linear decision functions. The key to this significance is the use of kernels, as was done with Parzen windows and the non-linear Perceptron [6]. A thorough study on kernel methods can be found in [13]

In this paper we initially introduce a unifying framework to describe these "classical" and show that they are different stages of refinement of the same decision function ; the only difference across is the way the training set itself is classified: Parzen windows do not require that the decision function separate correctly the

M.R. Berthold et al. (Eds.): IDA 2003, LNCS 2810, pp. 187–197, 2003.

training sets, the non linear Perceptron will be seen as a version of Parzen windows that requires that the training set be properly classified, while SVM will be seen as a version of the non linear Perceptron that gives an optimal solution to that classification. From this unifying framework we gain two major insights: the first is that all these algorithms, Parzen windows, SVM, non linear Perceptron, SBC, and arguably other algorithms based on the concept of maximizing similarity, assuming appropriate normalization of the data, converge to the same solution, as the "tightness" of the similarity measure increases. This property is established at the mathematical level and verified empirically. The other main insight we derive is the SBC algorithm, which in a sense subsumes these three algorithms and is an answer we propose to the open problem of finding a suitable "direct" multi-class SVM [2]. This is because we replace the concept of maximal margin, which is essentially a binary concept, by a concept of robustness of the decision function which is independent of the number of classes (but equivalent to the maximal margin in the binary case).
This paper discusses the issues of underfitting, overfitting, power of generalization, computational complexity and robustness within a simple framework using basic definitions of similarity matrix and tightness of similarity measures. We also compare at the mathematical level, the asymptotic behavior of these various algorithms and show that they converge with increasing tightness, assuming some normalization of the data, towards the same solution: the Maxsim formula. The SBC algorithm, derived from this study, uses naturally introduced similarity functions (not requiring mappings in high-dimensional Hilbert spaces), and optimizes "a la" SVM but without the need to solve the problem in a dual space with a complex quadratic objective function and more importantly with the ability to handle large multi-class problems. All this comes with a small penalty in the binary case, as the SVM decision function is smaller than the corresponding SBC function. In a concluding section we report briefly on an empirical validation of our claims, a full study is the object of a forthcoming paper.

2 Similarity Matrix

As in [13] we introduce the problem of classifying an object X by comparing its similarity to sets of previously classified training objects; X will be assigned to the class whose objects are most similar to X.
Given a set of I class-labeled training objects $\{X_i, \xi(X_i)\}$, $i = 1..I$, where $\xi(X_i)$ is the class of X_i, and for an unclassified object X, we define the *class similarity* of X with respect to a class C as

$$S_C(X) = \sum_{X_k \in C} \alpha_k s(X_k, X) \tag{1}$$

Where s is the similarity function and $\alpha_k \geq 0$ reflects the relative importance given to each X_k with respect to the classification.
We can therefore predict the class of X using the following decision function:

$$\xi(X) = arg_C\{max(S_C(X))\} \tag{2}$$

Or the strong version, which requires that not only X be more similar to class C than it is to any other class, but also be more similar to class C than it is to the union of any other collection of classes.

$$\xi(X) = arg_C\{max(S_C(X) > \sum_{D \neq C} S_D(X))\} \tag{3}$$

In order to compare our approach with classical Machine Learning algorithms that deal with binary classification, we will also consider the problem of supervised classification of objects from only two different classes A and B. In this case, (1) can be rewritten as:

$$S_A(X) - S_B(X) > 0 \rightarrow \xi(X) = A$$
$$S_A(X) - S_B(X) < 0 \rightarrow \xi(X) = B$$
$$S_A(X) - S_B(X) = 0 \rightarrow \xi(X) \text{ is } \textit{not defined}$$

Let α be the vector of coefficients α_i. The problem of a classification algorithm is to choose the criteria to compute these coefficients. In that respect some algorithms will be satisfied with finding a simply empirically acceptable solution, as Parzen windows do, while others will want a solution that satisfies predefined criteria, as the Perceptron does with a requirement of linear separability, and finally others, such as Support Vector Machines will find an optimal solution to these criteria. But all share a decision function with the structure of (1).

Among the many possible similarity measures, we will use mainly Radial Basis functions and polynomial kernels. This is because they are among the most popular and efficient, and also because they are the main ones used in SVM and the non-linear Perceptron, making comparisons more meaningful.

The Radial Basis Function (RBF in short): $s(x, y) = e^{\|X-Y\|^2/2\sigma^2}$ can be viewed as a similarity measure between two points in $\Re^n$, as it is a measure of the inverse of the distance between two points, its range is the interval (0,1) instead of an infinite range.

Depending on the applications, the data is represented by points in $\Re^n$, or normalized vectors. In this last case we consider bs; that is similarities based on a measure of the angle between two vectors of length 1. Those are quite suitable for many applications, such as text processing [7]

In any case it is easy conceptually, and trivial algebraically, to approximate a hyperplane by a spherical area, given a radius of sufficient magnitude. We just need to add a value corresponding to the radius of the sphere to the list of coefficients of each point, and normalize. So we can still address general classification problems of points in $\Re^n$, where we use vectors of uniform length to approximate the points.

Representatives of similarity measures for vectors of uniform length are the polynomial functions: $s(X, Y) = \|X, Y\|^n$ and of course the RBF.

We will say that similarity measure s is tighter than similarity measure t iff $\forall\ i, j \in I$, $s(X_i, X_j) \leq t(X_i, Y_j)$. The *similarity matrix* is defined as:

$$S = [\delta s(X_i, X_j)] \tag{4}$$

Where $\delta = 1$ if $\xi(X_i) = \xi(X_j)$, and $\delta = -1$ otherwise.

3 Fitting Conditions

We would like our decision function to satisfy two "fitting conditions", the first one simply expresses that the training vectors are properly classified. The second deals with parameterized measures, like RBF's σ parameter and the polynomial kernel's degree, which allow the user to control tightness. The condition essentially reflects the fact that extreme values of these parameters correspond to extreme tightness and overfitting, or no tightness at all and underfitting.

3.1 No Underfitting Condition

$$S\alpha > 0, \alpha \geq 0 \; is \; solvable \tag{5}$$

As each row of the system expresses that a training vector's similarity to its own class is superior to its similarity to the union of all other classes, we have:

Proposition 1: *The strong decision function correctly classifies the training vectors if and only if the set of constraints in (5) is solvable.*
We will say that a solution α (fits) the training data if and only if it satisfies the (no underfitting condition).

3.2 Absolute Over/Under Fitting Condition

We assume that the similarity functions are parameterized using a *tightness parameter q*, which allows the user to control the tightness of the similarity measure, and have the following properties:

$\exists\ q,\ \forall\ i, j \in I / S_q(X_i, X_j) = 1$
$\exists\ q, \forall i, j \in I / S_q(X_i, X_j) = 1 \iff i = j \wedge S_q(X_i, X_j) = 0$ for $i \neq j$

At one extreme setting the parameter q will make all objects equally similar. This corresponds to a situation where we have no means of comparing the objects, and leads to a decision function that assigns an object systematically to the class with the largest number of objects. This case we call absolute underfitting. At the other extreme q will make the similarity function become tighter and tighter until it becomes the identity step function. As the similarity function converges towards the identity step function, the decision function converges towards absolute overfitting; that is it will classify correctly only the training objects.
Both conditions are met by the RBF, or by polynomial kernels when the data has been normalized to vectors of length one.
The motivation for this condition, which is not fundamentally restrictive in nature, is that we find that the similarity matrix, which is a square symmetric matrix, has interesting theoretical and computationally advantageous properties as the tightness of the similarity measure increases:

Proposition 2: *As the tightness of the similarity measure increases, the similarity matrix becomes positive definite and consequently invertible.*
Proof (sketch):
We use Rayleigh's conditions and assume for sake of simplicity that the vectors are not duplicated. As the tightness increases the matrix S converges to the identity matrix and therefore min $\frac{\|SX,X\|}{(\|X\|\|Y\|)}$, the smallest eigenvalue, is strictly positive as the vector SX will be close to the vector X.

3.3 The Tight Fitting Zone

As the tightness increases the matrix S converges to the identity matrix and we have absolute overfitting. Then, of course the no underfitting condition becomes satisfied, as the unit vector (all coefficients equal to one) is a trivial solution to the system $I\alpha > 0,\ \alpha \geq 0$.
Now $\exists\alpha, S\alpha \geq 0$ is equivalent to stating that the vectors which are the rows of the matrix S are all on the same side of the hyperplane Ψ whose normal vector is α. As the tightness increases these vectors tend towards the vectors that form a basis of the vector space. Clearly they will pass, in general, on the same side of the hyperplane Ψ, defined by the unit vector, well before they reach the basis vectors. Furthermore, we also see that the unit vector provides a solution which is more and more b the basis. As a consequence we will say that a similarity measure is *tight fitting* when the unit vector is a solution to the no underfitting condition.
This brief analysis suggests that a similarity measure will be tight fitting, and consequently will admit a trivially computed decision function that satisfies the no underfitting condition, well before it becomes overfitting. This analysis also suggests that as we have data of increasingly complex shape/distribution, we have to use similarity measures of increased tightness, and as a consequence move closer to overfitting, but then the unit solution becomes closer to an optimal solution. This claim will be justified in later sections. It is an important claim as it essentially states that as the structure of the data becomes increasingly complex, a straightforward solution becomes more acceptable.
This solution is represented by the MaxSim (for Maximum Similarity) formula, which is the decision function 1 with the unit vector:

$$\xi(X) = arg_C\{max(\sum_{k\in C} S(X_k, X))\} \tag{6}$$

4 Asymptotic Behavior and Maximum Similarity

We proceed to formally establish how the MaxSim formula is related to some classification algorithms that we have used here as motivating examples. Specifically, we will show that the behavior of these algorithms converge asymptotically to equation (6)

4.1 Parzen Windows

The connection with Parzen windows is immediate, as the Maxsim formula (6) is a clear generalization of Parzen windows, and, as a corollary to proposition 2 we can see that Parzen windows become closer to SBC as the tightness of the similarity measure increases:

Proposition 3: *As the tightness of the similarity measure increases, Parzen windows will satisfy the no underfitting condition.*

4.2 Non-linear Perceptron

The Perceptron algorithm dates from 1956, designed by Rosenblatt at Cornell University. It provided the algorithmic foundation for the area of Neural Nets.
As opposed to the notion of maximum similarity, the Perceptron uses the notion of linear separability, which we briefly describe: The issue is to find a hyperplane Ψ separating two clouds of points in Rn, assuming that they are separable. Ψ will then be used as a decision function; a new point is assigned to the cloud which is on the same side of the hyperplane as the point.
There is a very elegant proof due to Novikov showing that when the two sets are separable the algorithm is guaranteed to find a solution, see [5] where the authors stress the significance of a dual view of the Perceptron. We use here this duality to show that the Perceptron satisfies the framework that we introduced in the previous section.
We know that the solution to the Perceptron algorithm can be expressed as $M^t\alpha$, which is as a linear combination of input vectors; from this remark and assuming a normalization of the input data, one can show:

Proposition 4: (Solving the system $MX > 0$ is equivalent to solving its dual $MM^t\alpha > 0$. The notion of linear separability is equivalent to the notion of no underfitting).
MM^t is a similarity matrix with the dot product as similarity measure. This means that the Perceptron algorithm classifies a test point according to a weighted sum of its similarities with the training points.

Corollary Replacing the dot product by an appropriate similarity measure we generalize the dual form of the Perceptron algorithm to a non linear multi-class version whose decision function converges to the Maxsim formula as the tightness of the similarity measure increases.

4.3 Support Vector Machines

MAXSIM and *SVM* are classifiers of a different nature. In the former all vectors play an equal role, in the latter as many as possible are eliminated. It does not seem evident that they could have similar behavior. However [6] have shown that the non-linear Perceptron can give results that are similar in accuracy to the SVM. Our version of the non-linear Perceptron might therefore also give

similar results. Furthermore [7] in his study of SVM for large scale applications indicates that a number of support vectors have their coefficients α_i equal to their limit C. It is also clear mathematically, and verified in practice, that when one increases the dimension (by using kernels of greater non linearity), the number of support vectors increases. It seems that with increasing complexity of the data, and therefore increasing non linearity, the number of coefficients α_i that take non-zero values increases, which seems to indicate that the behavior of the SVM in such cases tends towards the behavior of the Maxsim formula.
We will give a formal proof of this, under the following conditions:

- The training vectors are all different (no duplicates)
- The training vectors have been normalized to length one.
- The kernels are polynomial and we consider increasing powers, or the kernels are radial Basis Functions and we consider decreasing O in which case the preceding normalization is not compulsory.
- The two subsets of the training set that have to be separated should have the same size.

These restrictions do not affect the generality of our argument, as we know that the existence of duplicate vectors can affect the execution of the SVM, making the matrix singular, the normalization is done in many applications, and can be done easily in all cases, the kernels mentioned are used in most applications, and the property we will use may well be shared with other kernels. We require that the two subsets have the same size to make the presentation clearer, it is not needed otherwise.
Under these conditions we will establish the following proposition:

Proposition 5: *In the limit of non-linearity, the decision function of Support Vector Machines converges to the Maxsim formula.*

Proof (sketch):

There are a number of equivalent mathematical formulations of the SVM, we will choose one from [3] and [4].

$$\begin{gathered} min(\alpha^t Q \alpha) \\ 0 \leq \alpha_i < 1 \\ e^t \alpha = \nu l \\ y^t \alpha = 0 \end{gathered}$$

And the decision function is:

$$sgn(\sum y_i \frac{\alpha_i}{\rho}(k(X_i, X) + b)) \tag{7}$$

Under our hypothesis, the matrix Q is equal to a similarity matrix $[\delta k(X_i, X_j)]$. For polynomial kernels with degrees tending to infinity, or for a RBF whose parameter σ tends to 0, the matrix Q converges to the *identity matrix*.

The solution of the SVM is then the point on the plane $e^t \alpha = \nu l$, whose distance to the origin is minimum, that is a point such that all coordinates α_i are equal, and as the two subsets have the same number of vectors, the last equality is trivially verified.

5 The SBC Algorithm

As a consequence of Proposition 2, we know that by choosing a tight enough similarity measure, the No Underfitting condition will be satisfied. We also know that the unit vector may or may not provide a solution, but even if it does, there will be infinitely many weight vectors to choose from. As we have seen in the last section, Parzen Windows and Non-linear perceptrons are only two possible choices for these vectors; on the other hand, SVM provides an optimal solution by finding a maximal margin hyperplane. In our case, we also would like to find a solution which is optimal in some sense. We will justify our proposed optimization in two ways:

- Robustness: we want the decision function to give similar values for two vectors of similarities that are very close to each other (similar points should be similarly classified). So its values should not vary too steeply.
- Power of generalization: the less it varies (and provided the no underfitting conditions are met), the more points can be safely assigned to any given class.

As a consequence we want to minimize the norm of the gradient, which controls the variations of the values of the decision function. We give now a more geometric justification, which also shows a link with the SVM's choice of maximizing the margin.
First, we replace the condition $S\alpha > 0$ by the equivalent $S\alpha > 1$.
We also assume, for sake of simplicity, that the rows of S have been normalized to length 1.
A row of the system now reads as:

$$\cos(\theta_{i\alpha}) > \frac{1}{\|\alpha\|} \tag{8}$$

Where $\theta_{i\alpha}$ is the angle between the ith row vector and the weight vector α. A robust solution is one for which small changes in the weight vector will still give a weight vector solving the system. Therefore a more robust solution is achieved when one minimizes $\|\alpha\|$. These considerations lead us to the following quadratic program, which determines the values of the weight vectors of the decision function, which we call SBC:

$$\begin{gathered} min(\|\alpha\|^2) \\ S\alpha > 1 \\ \alpha \geq 0 \end{gathered}$$

5.1 Implementation Issues

This quadratic program has many attractive features from a computational point of view. The objective function is a very simple convex function, the underlying quadratic form uses the identity matrix, and according to proposition 2, as the

tightness increases, the system of linear constraints admits a trivial solution (the unit vector). So it will run well with standard quadratic programming software such as MatLab, but there is much scope for more efficient implementations. For instance, because the objective function is convex, one can use the Karush-Kuhn-Tucker conditions to compute the optimal solution via a simple modification to the simplex algorithm [10]. Also, we can remark that the program computes the distance of a polyhedral set to the origin, so one could derive an algorithm based on such geometrical consideration, in a way inspired by [8], which computes the maximal margin as the distance between two polytopes.
Finally, as with "pure" SVM, this "pure" version of SBC is sensitive to noisy data, even though it performs quite well on a number of simple KDD benchmark data (see next section). This problem can be taken care of by adopting SVM's solution(s) to this problem [2] [5], for instance introducing a parameter C to control the amount of allowed training data misclassifications.

6 Empirical Validation

We have chosen to compare experimentally the Maxsim formula and our more refined SBC algorithm with Chang and Lin's implementation of Support Vector Machines, LIBSVM (version 2.33) [3]. It is based on the formulation of the quadratic programming optimization that was our basis for a theoretical comparison, but also it is an excellent implementation of SVM which won supervised learning competitions, it also provides software for multi-class, density function and other extensions of the SVM technology. We have chosen data sets which are benchmarks in machine learning for classification and were found on the Silicon Graphics site http://www.sgi.com/tech/mlcdb/index.html, which contains a wealth of information about the origin, format, uses of these sets, and also accuracy comparisons of various machine learning algorithms. It is worth mentioning that SVM edged out the other methods in most of the tests sets we used, which is the reason we only considered SVM for comparison. Also, the reported results in each case are the best we could obtain by trying different sets of parameters.
We provide here only a summary, as a complete and detailed treatment including implementation issues and an analysis of the statistical significance of the differences in accuracy is to be found in a forthcoming paper.
Surprisingly, the best fit between the Maxsim formula and the SVM was found for small data sets: they achieved the same accuracy, up to two digits: 96.67% for the wine recognition data; for the Pima Indians diabetes data, they both achieved 75%. Slight differences were found for the liver disorder data: for polynomial kernels the SVM achieved 73.91% accuracy, while the Maxsim formula achieved 77.39%, while with RBF kernels, the SVM achieved 79% and the Maxsim formula 77%. Similarly, for the satellite image recognition data, the SVM achieved 92% to 91% for Maxsim. For the letter recognition problem, Maxsim achieved 96.4% and SVM achieved 94.8% with polynomial kernels and 95.6% and SVM 96.7%, with RBF.

The SBC gave either exactly the same results as the SVM, or slightly better.
We also have experimented with the classification of biological signals [11]. It is known that present day data bases of genomic and proteomic data contain a large number of errors (sometimes up to 25%), or hypothetic information solely verified by visual inspection of patterns. In order to avoid interference of noisy or unreliable data with our comparison, we have used data whose quality has been experimentally confirmed. We performed two experiments: splice site detection with the data set obtained from [14] and E. Coli start sites prediction with the dataset obtained from [9] and stored in ECOGENE [12]. We found out by cross validation that Maxsim, SVM and SBC give excellent results in the range of 92% to 100% depending on the sets, Maxsim was consistently slightly lower or equal to SVM, while the SVM was consistently slightly lower or equal to SBC. However for more noisy problems, the SBC has to be implemented with the same considerations as the SVM; that is with a C parameter which controls noise and avoids overfitting.
Finally, we also compared the SVM and Maxsim when given a large multi-class problem. For this we derived a problem to recognize two-letter words. That is, we have now 676 classes instead of 26 (as in the original letter-recognition problem). SVM took 17'01" to run, while Maxsim took 42" ; this difference is due to the quadratic complexity of both training and testing phases of the SVM on the number of classes.

7 Conclusion

We have described a general framework for the study of kernel-based machine learning classification algorithms in a simple and systematic way. We also have shown that these algorithms converge to the Maxsim formula (4.3), as the tightness increases. In practice, the Maxsim formula seems to give another solution 'in the middle' with the added capability of handling multi-class problems in a natural way. The SBC algorithm, described in section 5, also handles multi-class problems naturally and more importantly it gives a solution which is optimal. The experiments confirmed this, as the performance of SBC was at least as good as the SVM in all cases. We believe our main contribution is that we accomplished SVM-like accuracy numbers while maintaining a transparent and efficient handling of multi-class problems.

References

1. MPS Brown WN Grundy, D Lin, N Cristianini, CW Sugnet, TS Furey, M Ares, D Haussler.: Knowledge-based analysis of microarray gene expression data by using support vector machines. Proceedings of the National Academy of Sciences 97, 262–267, 2000.
2. Burges, C. J. C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):121–167. 1998
3. Chang C. C., and C. J. Lin.: Training ν-support vector classifiers, Theory and Algorithms. Neural Computation (13) 9 2119–2147, 2001.

4. Crisp D.J., and C.J.C. Burges 2000. In: S. Solla, T. Leen and K.R. Muller (Eds). A Geometric Interpretation of N=bsvm Classifiers. Advances in Neural Information Processing Systems Vol 12 Cambridge MA MIT Press, 2000.
5. N. Cristianini and J. Shawe-Taylor.: An Introduction to Support Vector Machines. Cambridge University Press, 2000.
6. Freund Y., and R.E. Schapire.: Large Margin Classification using the Perceptron Algorithm. Machine Learning (37) 3 277–296, 1999
7. T. Joachims.: Learning to Classify Text using Support Vector Machines.: Kluwer, 2002
8. S.S. Keerthi, S.K. Shevade, C. Bhattacharyya and K.R.K. Murthy.: A fast iterative nearest point algorithm for support vector machine classifier design. IEEE Transactions on Neural Networks, Vol. 11, pp.124–136, Jan 2000.
9. Link, A.J., Robinson, K. and Church, G.M.: Comparing the predicted and observed properties of proteins encoded in the genome of Escherichia Coli. Electrophoresis., 18, 1259–1313, 1997
10. K. G. Murty.: Linear Complementarity, Linear and Nonlinear Programming.: Helderman-Verlag, 1988.
11. Martin G. Reese, Frank H. Eeckman, David Kulp, David Haussler.: Improved Splice Site Detection in Genie. RECOMB. Santa Fe, ed. M. Waterman, 1997.
12. Rudd, K. E.: Ecogene: a genome sequence database for Escherichia Coli K-12. Nucleic Acid Research, 28, 60–64, 2000
13. B Scholkopf, A Smola.: Learning with Kernels. MIT Press, 2001
14. T.A. Thanaraj.: A clean data set of EST-confirmed splice sites from Homo sapiens amnd standards for clean-up procedures. Nucleic Acids Research. Vol 27, No. 13 2627–2637, 1999