

# Pattern Classification with Polynomial Learning Machines

Jonathan P. Bernick

Department of Computer Science  
Coastal Carolina University  
P. O. Box 261954  
Conway, South Carolina 29528, USA  
[jbernick@coastal.edu](mailto:jbernick@coastal.edu)

**Abstract.** *We present a new kind of pattern classifier, the Polynomial Learning Machine (PLM). The PLM is derived, and its construction detailed. We compare the performance of the PLM to that of the Support Vector Machine (SVM) in classifying binary-classed data sets, and find that the PLM consistently trains and tests in a shorter time than the SVM while maintaining comparable classification accuracy; in the case of very large data sets, the PLM training and testing times are shorter by orders of magnitude. We conclude that the PLM is a useful tool for pattern classification, and worthy of further investigation.*

**Keywords:** pattern classification, multi-layer perceptron, support vector machine, neural network, pattern recognition

## 1. Introduction

Pattern classification has long been a topic of interest to the machine learning community. From early neural networks such as multi-layer perceptrons, to modern learning machines such as support vector machines (SVM's), enormous effort has been put into developing learning machines that can correctly identify a vector of input values as belonging to one of a finite set of classes ([2], [3], [5]).

In this paper, we present a new kind of pattern classifier, the *Polynomial Learning Machine* (PLM). Though still in the early stages of development, the PLM has demonstrated an ability to reliably generalize using difficult standard sets, and it is highly probable that this capability can be increased through further investigation. In particular, we compare the performance of the PLM to the SVM, and find that the PLM performs pattern classification with an accuracy comparable to that of the SVM, and a speed consistently greater than that of SVM, often by several orders of magnitude.

## 2. Theory

### 2.1 Definition

A PLM is a binary classifier represented by a set of values  $\{M, D, f\}$ , where

1.  $M$  is the number of elements in the input vectors to the PLM.
2.  $D$  is the degree of the polynomial membership function.
3.  $f: \mathcal{R}^M \rightarrow \mathcal{R}$  is the polynomial membership function.

The output of a PLM for an input vector  $\mathbf{x}$  is the class whose membership function has the largest value for input  $\mathbf{x}$ ; i.e.,

$$O(\mathbf{x}) = \begin{cases} 1 & f(\mathbf{x}) \geq p \\ 0 & f(\mathbf{x}) < p \end{cases}, \quad (1)$$

where  $p$  is some real threshold value, and

$$f(\mathbf{x}) = a_0 + a_1x_1 + \dots + a_Mx_M + a_{(M+1)}x_1^2 + a_{(M+2)}x_1x_2 + \dots + a_{E0-2}x_{M-1}^D + a_{E0-1}x_M \quad (2)$$

and  $E0$  may be calculated from the recursive function

$$\begin{aligned} E0(M,D) &= E0(M-1, D) + E0(M, D-1), \\ E0(M, 1) &= M + 1, \\ E0(0, D) &= 1. \end{aligned} \quad (3)$$

## 2.2 Derivation

Let us assume there exist two mutually-exclusive finite sets of  $M$ -element vectors  $V_0, V_1$ , and a function  $g$  such that for any  $\mathbf{x} \in \cup\{V_0, V_1\}$

$$g(\mathbf{x}) = \begin{cases} b & \mathbf{x} \in V_0 \\ -b & \mathbf{x} \in V_1 \end{cases}. \quad (4)$$

for some positive  $b$ .

It is well-known that a function  $f(\mathbf{x})$  that is continuous, bounded, and infinitely differentiable in the neighborhood  $(\mathbf{c} - \mathbf{b}, \mathbf{c} + \mathbf{b})$  of a point  $\mathbf{c} = (c_1, c_2, \dots, c_M)$  for some finite  $\mathbf{b} \in \mathfrak{R}^M$  may be approximated as a polynomial in that neighborhood by the Taylor series expansion

$$f(\mathbf{x}) \approx f(\mathbf{c}) + \sum_{i=1}^D \left( \frac{1}{i!} \left[ \prod_{j=1}^i \left( \sum_{k=1}^M (x_k - c_k) \frac{\partial}{\partial q_k} \right) \right] f(q_1, \dots, q_M) \right) \Bigg|_{(q_1, \dots, q_M) = (c_1, \dots, c_M)} \quad (5)$$

with the difference between the left and right sides of (5) going to zero as the degree  $D$  of the Taylor series polynomial goes to infinity. If we now define  $T_d(\mathbf{x})$  as the Taylor series expansion of  $g(\mathbf{x})$  of degree  $D$  about  $\mathbf{x} = \mathbf{0}$  (also known as the *Maclauren series*), we may thus rewrite (4) as

$$g_i(\mathbf{x}) \approx f(\mathbf{x}) = T_d(\mathbf{x}), \quad (6)$$

which gives us the form of the membership functions in (2). Since  $f(\mathbf{x})$  is an approximation of  $g(\mathbf{x})$ , it is unlikely that  $f(\mathbf{x})$  will go exactly to  $b$  for  $\mathbf{x} \in V_0$ ; however, given that  $f(\mathbf{x}) \approx -b$  if  $\mathbf{x} \in V_1$ , it is reasonable to expect that  $f(\mathbf{x})$  will have a value closer to  $b$  than  $-b$  if  $\mathbf{x} \in V_0$ . Accordingly, we deduce the behavior of the PLM as direct implication of modeling membership functions using the Taylor series approximation.

## 2.3 Construction

Consider the case of training a PLM of degree  $D$  using a set of  $K$  training pairs of the form  $(\mathbf{x}, y)$ , each of which is contained in exactly one class  $V_0$  or  $V_1$ , where  $y$  is the index of the class of  $\mathbf{x}$ , with  $D$  chosen such that  $E0 < K$ . Let  $\mathbf{x}_j = [x_1, \dots, x_M]$  denote the  $j$ th input vector and  $y_j$  the class of  $\mathbf{x}_j$ ; if we define

$$\mathbf{u}_j = [1, x_1, \dots, x_M, x_1^2, x_1x_2, \dots, x_{M-1}^D, x_M^D] \quad (7)$$

and

$$\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K]^T, \quad (8)$$

we may then form the linear system

$$\mathbf{U}\mathbf{a} = \mathbf{b}, \quad (9)$$

where

$$\mathbf{b} = [b_1, \dots, b_K], \quad (10)$$

$$b_i = \begin{cases} 0 & \mathbf{u}_i \in V_0 \\ 1 & \mathbf{u}_i \in V_1 \end{cases}.$$

and  $\mathbf{a}$  is the vector of the coefficients of the polynomial  $f(\mathbf{x})$ .

Let us assume without loss of generality that one of our classes has more members than the other. We now wish to weight our system so that those training pairs whose input vectors are members of the smaller class will have approximately the same influence on the value of  $\mathbf{a}_i$  as those that are members of the larger class. This is accomplished by left-multiplying both sides of (9) by a  $K \times K$  diagonal weight matrix  $\mathbf{W}$ . A first approximation for  $\mathbf{W}$  may be created such that, if  $K_i$  out of  $K$  input vectors are members of the smaller class, then the trace elements  $W_{jj}$  are chosen such that

$$W_{jj} = \begin{cases} \frac{K}{K_i} - 1 & y_j = i \\ 1 & y_j \neq i \end{cases}, \quad (11)$$

although for maximum performance  $\mathbf{W}$  will have to be determined empirically. (We emphasize that (11) is a heuristic, and the development of an algorithm to construct  $\mathbf{W}$  to maximize the percentage of vectors classified correctly by a PLM is an open and interesting research problem.)

Once  $\mathbf{W}$  is known, we multiply both sides of (9) by  $\mathbf{U}^T \mathbf{W}$ , giving us

$$\mathbf{U}^T \mathbf{W} \mathbf{U} \mathbf{a} = \mathbf{U}^T \mathbf{W} \mathbf{b}. \quad (12)$$

One final left-multiplication (this time by the inverse of  $\mathbf{U}^T \mathbf{W} \mathbf{U}$ ), and we obtain the error-minimizing least-squares solution of an overdetermined system

$$\mathbf{a} = (\mathbf{U}^T \mathbf{W} \mathbf{U})^{-1} \mathbf{U}^T \mathbf{W} \mathbf{b}, \quad (13)$$

which gives us the polynomial

$$f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{u} \rangle, \quad (14)$$

where  $\mathbf{u}$  is the expansion of  $\mathbf{x}$  as per (7).

We note that the calculation in (13) poses some interesting computational problems. Since matrix multiplication and inversion are in general  $O(n^3)$  processes, the amount of time required to compute  $\mathbf{a}_i$  grows drastically as  $D$  increases. There are several ways to mitigate this growth, including

1. Use the smallest possible value of  $D$ . This is also desirable for other reasons, which are discussed later in this paper.
2. Since  $\mathbf{U}^T \mathbf{W} \mathbf{U}$  is symmetric and positive-definite [7], the Cholesky decomposition may be used to make the substitution  $\mathbf{U}^T \mathbf{W} \mathbf{U} = \mathbf{L} \mathbf{L}^T$ , where  $\mathbf{L}$  is a lower-triangular  $E_0 \times E_0$  matrix, approximately halving the time required to compute the inversion of  $\mathbf{U}^T \mathbf{W} \mathbf{U}$ .

## 2.4 Stability and Range

It can easily be proven that the degree  $D$  of a PLM acts as a limit both on the intricacy of the function that may be closely approximated by that PLM and the PLM's potential for erratic behavior.

**Theorem.** Let  $X = [p_1, q_1] \times \dots \times [p_M, q_M]$  be the input space for a PLM  $P\{M, D, f(\mathbf{x})\}$ . As the value of input vector element  $x_i$  increases from  $p_i$  to  $q_i$ , or decreases from  $q_i$  to  $p_i$ , the output of  $P$  may change value at most  $D$  times.

**Proof.** Let us allow the value of  $x_i$  to vary while holding the values of all other input vector elements constant. Under such circumstances, any change in the value of  $f(\mathbf{x})$  will be solely dependent upon change in the value of  $x_i$ . Since such a polynomial may have at most  $D$  discrete roots, the value  $f(\mathbf{x})$  may cross zero at most  $D$  times in the range  $[p_i, q_i]$ . If we assume without loss of generality that the threshold  $p$  of  $P$  is zero, it is obvious that the output of  $P$  may also change at most  $D$  times.

The implications of this proof are twofold. Firstly, it suggests that there can be no chaotic regions in the input space of a PLM, since the number of transitions the output of the PLM may undergo over that region is finite. Secondly, it implies that in using the PLM to model a function with many critical points some detail will be lost (in practice, this has proven itself not to be a problem, as the next section will show).

## 3. Results and Discussion

### 3.1 Overview

To test the performance of the PLM at pattern classification versus that of the SVM, implementations of both algorithms were used to classify several benchmark data sets, and the results compared. The Radial Bias Function (RBF) SVM was used instead of the linear SVM because the former is generally more accurate when classifying data [1]. Typical results may be found in Table 1.

### 3.2 Data Sets

For the most part, the data sets used were standard benchmark data sets obtained from the SGI/UCI Data Repository [6] and used without modification. The exceptions were:

1. The LDSH, LDSW, and LDST training and test data sets were created by taking the Cartesian join of an existing set of vectors (from Haberman Survival Data, Wisconsin Breast Cancer Database, and Tic-Tac-Toe respectively), with the target of a joined vector being the XOR of the targets of the participating vectors. In each case, training data was joined with training data, and test data with test data.
2. The original Connect-4 data set had trinary targets (win/lose/draw). To create binary data sets, the target for draw was first made the same as the target for win (resulting in the lose/not lose data set) and then lose (resulting in the win/not win data set).
3. The 19-Bit Majority Function data was generated by the author using a simple program.

### 3.3 Software and Equipment

The PLM was modeled using a C++ program written and compiled using Microsoft Visual Studio 6.0. The LIBSVM package was used to simulate the SVM [4].

In all cases except the LDSW, LDST and 19-Bit Majority Function, the PLM and SVM simulations were run on a Hewlett-Packard laptop with 512 MB of memory and a 1.5 GHz Celeron M 370 processor, while the LDSW, LDST and 19-Bit Majority Function PLM and SVM simulations were run on two Dell desktop computers, each of which had with 1 GB of memory and a 1.7 GHz Xeon processor. For all data sets, both the PLM and SVM simulations using a given set were run on the same computer.

### 3.4 Results

It is apparent from the results in Table 1 that the PLM consistently both trains and tests data faster than the SVM. This difference is particularly striking in the cases of the two largest data sets (408321 and 524228 training vectors, respectively), where the PLM was able to train in a few minutes (and test in a small fraction of those minutes) while the SVM had not reached a solution after several days.

Throughout the results the PLM and SVM produced similar levels of classifying accuracy. The only major discrepancies are in the Tic-Tac-Toe and Connect-4 data sets: In the former case, the SVM overfitted; in the latter case, round-off errors prevented the use of a higher-degree PLM, which would have increased accuracy.

The results obtained here suggest that the PLM gives its best performance when classifying large data sets with relatively few attributes; we shall expand upon this observation in the next section. Also, we note that in no case was a PLM of degree larger than 2 necessary to classify any data set; given the rapid growth of the size of matrix  $U^T U$  with respect to the value of the degree of the PLM polynomial, this reassures us that difficult-to-classify data sets can be modeled in a reasonable amount of time.

Data Sets				PLM					SVM – RBF			
Name	Train. vect.	Test. vect.	Attr.	Degree	Weight	Class. %	Train time (sec.)	Test time (sec.)	$\gamma$	Class. %	Train time (sec.)	Test time (sec.)
19 – Bit Majority Function	524288	--	19	1	1.000	<b>100.0</b>	659	0.203	1.0	?	> 72 hours	--
Large Data Set – Tic-Tac-Toe	408321	101761	18	2	1.783	<b>85.86</b>	1940	11.98	1.0	?	> 72 hours	--
Large Data Set - Wisconsin	207936	51529	18	2	1.000	<b>97.34</b>	982	0.219	0.5	<b>96.96</b>	4522	1942
Connect-4 (lose/not-lose)	50669	16889	42	1	1.010	<b>83.26</b>	16.63	0.015	1.5	<b>90.98</b>	1812	706
Connect-4 (win/not-win)	50669	16889	42	1	1.016	<b>79.27</b>	23.91	0.016	0.8	<b>89.50</b>	1305	771
Large Data Set - Haberman	41616	10404	6	1	1.360	<b>69.98</b>	6.062	0.000	2.0	<b>68.51</b>	412	302
<i>Agaricus Lepiota</i>	5416	2708	22	1	1.051	<b>96.79</b>	0.421	0.000	1.6	<b>100.0</b>	3.641	2.875
Spam E-mail Database	3068	1535	57	1	1.476	<b>91.60</b>	1.890	0.016	1.7	<b>92.89</b>	3.062	3.047
Chess End-Game - KRKPA7	2130	1066	36	1	1.420	<b>94.00</b>	0.390	0.000	0.75	<b>97.19</b>	1.781	1.75
Tic-Tac-Toe Endgame Database	639	319	9	2	4.313	<b>99.06</b>	0.031	0.000	1.500	<b>88.71</b>	0.140	0.204
Pima Indians Diabetes Database	512	256	8	1	1.481	<b>76.56</b>	0.016	0.000	0.60	<b>74.61</b>	0.109	0.203
Wisconsin Breast Cancer Database	456	227	9	2	3.730	<b>99.12</b>	0.062	0.000	0.005	<b>99.56</b>	0.078	0.141
House Votes 1984	290	145	16	1	14.68	<b>93.10</b>	0.015	0.000	0.44	<b>95.17</b>	0.062	0.109
BUPA Liver Disorders	228	115	6	2	0.863	<b>78.26</b>	0.015	0.000	2.575	<b>76.52</b>	0.062	0.109
Haberman’s Survival Data	204	102	3	2	1.112	<b>84.31</b>	0.000	0.000	15.0	<b>81.37</b>	0.062	0.094

**Table 1** – Comparison of PLM and SVM at classifying benchmark data sets. In all cases,  $p = 0.5$

Finally, fairness compels us to note that, had we been able to run more SVM simulations for the large data sets, it is possible that we would have obtained higher classification accuracies; however, the much-longer training times for the SVM made this impossible.

### 3.5 Discussion

There are numerous reasons to suggest that PLM is a useful complement to the SVM for pattern classification. The first is the difference in computational approaches: If  $\mathbf{U}$  is the matrix whose rows are the training vectors, training the SVM requires the computation (and possibly inversion) of outer matrix product  $\mathbf{U}\mathbf{U}^T$ , with result that the number of operations to train a SVM is of order  $n^3$  with respect to the number of vectors in the training set. With the PLM, on the other hand, the main matrix computations are the inner product  $\mathbf{U}^T\mathbf{U}$  and its inversion, with result that the number of operations to train a PLM is of order  $n^2$  with respect to the number of vectors in the training set. This implies that the PLM is the much faster tool with which to classify from a large training sets, and this implication is supported by the results in Table 1.

Additionally, the PLM is more robust in the face of certain types of noisy data than the SVM. Duplicate training vectors (with identical or different target values) will cause the SVM to fail as they result in the matrix product  $\mathbf{U}\mathbf{U}^T$  becoming singular; in contrast, the PLM treats duplicate training vectors as additional weighting of those vectors. The corresponding case with regard to the PLM is the unlikely event that  $\mathbf{U}$  has a column or columns which is/are linearly dependent on some other column(s) in  $\mathbf{U}$ ; in this case, though, the offending column or columns may be identified by means of Gaussian elimination on the columns of  $\mathbf{U}$ , and thus removed.

Furthermore, the use of the inner product  $\mathbf{U}^T\mathbf{U}$  allows multiple PLM's using different weightings to be produced in a shorter time per capita than a single PLM. Let matrices  $\mathbf{U}_1$  and  $\mathbf{U}_2$  denote those rows of  $\mathbf{U}$  contained in the larger and smaller classes respectively; if we then define

$$\mathbf{H}_1 = \mathbf{U}_1^T\mathbf{U}_1 \quad (15)$$

and

$$\mathbf{H}_2 = \mathbf{U}_2^T\mathbf{U}_2, \quad (16)$$

then simple algebra shows that

$$\mathbf{U}^T\mathbf{W}\mathbf{U} = \mathbf{H}_1 + w\mathbf{H}_2, \quad (17)$$

where  $w$  is a scalar weight. Since  $\mathbf{H}_1$  and  $\mathbf{H}_2$  are invariant with respect to  $w$ , the right hand side of (17) may be used in the place of explicit computation of  $\mathbf{U}^T\mathbf{W}\mathbf{U}$  for multiple weightings, greatly increasing the speed with which the value of this matrix is computed.

### 3.6 Open Research Issues

Several open research issues remain with regard to the PLM.

Firstly, it seems unlikely that all the columns of the data matrix  $\mathbf{U}$  of a PLM of degree 2 or greater are necessary for the classification of data. Finding an algorithm by which unnecessary columns could be identified and removed would have the potential to increase the speed with which a PLM could be trained.

Secondly, it would be highly desirable to have a generalization of the PLM from the binary case to the  $n$ -class case. While, as in the case of the SVM, ad-hoc methods exist (e.g. one-vs.-one, one-vs.-all), these methods add greatly to the training time of the PLM. In theory it should be possible to select target values so as to allow the PLM to handle multiclass problems via interpolations; however, finding an algorithm with which to perform such a selection has proven to be a nontrivial task.

Finally, the use of the inner product  $\mathbf{U}^T\mathbf{U}$  results in a dense matrix, which must then be inverted. As the size of  $\mathbf{U}^T\mathbf{U}$  increases roundoff error propagates rapidly in the inversion. If the PLM is to be used on data sets with large numbers of attributes, mathematical tools must be chosen from numerical analysis to counteract this effect.

## 4. Conclusion

In this paper we have presented the Polynomial Learning Machine. We have compared this learning machine to the SVM, and found that it consistently trains and tests more quickly than the SVM while maintaining comparable classification accuracy, particularly when the data set to be classified has a large number of training vectors with relatively few attributes. We conclude that the PLM is a useful tool with which to perform pattern classification on such data sets, and recommend further investigation and study.

## 5. Addendum

After the submission of this paper, an existing work ([8]) was discovered which duplicated some of results contained herein.

## 6. Acknowledgements

The author would like to thank Jean-Louis Lassez for our numerous enlightening conversations about the PLM, and the staff of the Myrtle Beach Waffle House (US 501) for providing the copious quantities of unsweetened iced tea that made the production of this paper possible.

## 7. References

- [1] Cristianini, N., and Shawe-Taylor, J., *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press; 1st edition, March 23, 2000.
- [2] Dong, J., Krzyzak, A., and Suen, C., "Fast SVM Training Algorithm with Decomposition on Very Large Data Sets," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 27, No. 4, April 2005.
- [3] Kim, H., Pang, S., Je, H., Kim, D., Bang, S., "Pattern Classification Using Support Vector Machine Ensemble," *Proceedings of the 16th International Conference on Pattern Recognition*, Quebec, Canada, August 11-15, 2002.
- [4] LIBSVM, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/> .
- [5] Liu, W., Shen, P., Qu, Y., and Zia, D., "Fast Algorithm of Support Vector Machines in Lung Cancer Diagnosis," *Proceedings of the International Workshop on Medical Imaging and Augmented Reality*, Hong Kong, June 10-12, 2001.
- [6] SGI/UCI Data Repository, <http://www.sgi.com/tech/mlc/db/> .
- [7] Strang, G., *Linear Algebra and Its Applications*, Academic Press, Inc., New York, 1980.
- [8] Toh, K.-A., Tran, Q.-L., Srinivasan, D., "Benchmarking a Reduced Multivariate Polynomial Pattern Classifier," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 26, No. 6, June 2004.