Nonparametric Classification with Polynomial MPMC Cascades ; CU-CS-955-03

Sander Bohte
University of Colorado Boulder

Gregory Grudic
University of Colorado Boulder

Follow this and additional works at: http://scholar.colorado.edu/csci_techreports

Recommended Citation
http://scholar.colorado.edu/csci_techreports/897
Nonparametric Classification with Polynomial MPMC Cascades

Sander M. Bohte\textsuperscript{1,2}  
S.M.Bohte@cwi.nl  
\textsuperscript{1}Dept. of Computer Science  
University of Colorado, Boulder, USA  
\textsuperscript{2}Dept. Software Engineering  
CWI, Amsterdam, The Netherlands

Gregory Z. Grudic\textsuperscript{1}  
grudic@cs.colorado.edu  
\textsuperscript{1}Dept. of Computer Science  
University of Colorado, Boulder, USA

Abstract

A new class of nonparametric algorithms for high-dimensional binary classification is presented using cascades of low dimensional polynomial structures. Construction of polynomial cascades is based on Minimax Probability Machine Classification (MPMC) [Lanckriet et al., 2002], which results in direct estimates of classification accuracy, and provides a simple stopping criteria that does not require expensive cross-validation measures. This Polynomial MPMC Cascade (PMC) algorithm is constructed in linear time with respect to the input space dimensionality, and linear time in the number of examples, making it an attractive alternative to computationally expensive algorithms like support vector machines and standard MPMC. Experimental evidence is given showing that, compared to state-of-the-art classifiers, PMCs are competitive; inherently fast to compute; not prone to overfitting; and generally yield accurate estimates of the maximum error rate on unseen data.

1 Introduction

The first goal of this paper is to propose a computationally efficient class of nonparametric binary classification algorithms that generate nonlinear separating boundaries, with minimal tuning of learning parameters. We want to avoid the computational pitfalls of using extensive cross validation for model selection. For example, in Support Vector Machines (SVMs) [1], both the choice of kernels and corresponding kernel parameters is based on extensive cross validation experiments, making generating good SVM models computationally very difficult. Other algorithms, such as Minimax Probability Machine Classification (MPMC) [2, 3], Neural Networks, and even ensemble methods such as Boosting [4], can suffer from the same computational pitfalls.

The second goal of this paper is to have the proposed class of algorithms give explicit estimates on the probability of misclassification on future data, without resorting to unrealistic distribution assumptions or computationally expensive density estimation [5]. The Minimax Probability Machine for Classification (MPMC), due to Lanckriet et al. [2, 3], is a recent algorithm that has this characteristic. Given the means and covariance matrices of two classes, MPMC calculates a hyperplane that separates the data by minimizing the maximum probability of misclassification. As such, it generates both a classification and a bound on the expected error for future data. In the same paper, the MPMC is also extended to non-linear separating hypersurfaces using kernel methods. However, as indicated above, MPMC is not computationally efficient, having similar complexity as SVM algorithms.
To address these two goals, we propose an efficient, scalable, nonparametric approach to generating nonlinear classifiers based on the MPMC framework: the class of Polynomial MPMC Cascades (PMCs). PMCs are motivated by the Polynomial Cascade algorithm for regression, due to Grudic & Lawrence [6], which efficiently builds very high dimensional, nonlinear regression surfaces using cascades of low dimensional polynomials. The proposed Polynomial MPMC Cascade algorithms generate a nonlinear hypersurface from a cascade of low-dimensional polynomial structures. The optimal choice for each level of the cascade is determined using MPMC to select the next most discriminating structure. From one level to the next, these additional discriminating structures are added to the cascade using MPMC, such that at each step we obtain the next most discriminating polynomial cascade; we construct PMC variants that use different ways of constructing the initial polynomial structures. By using MPMC to guide the addition of new cascade levels, we maintain a current performance and current maximum error bound during construction. We stop the addition of new structures to the cascade when the error bound no longer improves.

We show that the PMC algorithms yield very competitive results on benchmark problems, while providing maximum error bounds. The PMCs are highly efficient in that their complexity is 1) linear in the number of input-dimensions, 2) linear in the number of training examples, 3) linear in the number of levels of the cascade, and 4) cubed in the (low!) order of the polynomial structures (the levels). The nonparametric nature of the PMCs is exemplified by the fact that the one free parameter in the algorithm is the order of the polynomial structures. In this paper, we explore the simplest nonlinear structure: the quadratic polynomial.

To summarize, we believe that the contribution of this paper lies in effectiveness and speed of the proposed class of PMC algorithms: while being solidly rooted in the theory of MPMC, their linear complexity and nonparametric nature allow them to essentially be a “plug & play” solution for classification problems, yielding results competitive with highly computationally intensive algorithms like MPMC with Gaussian kernels and non-linear SVMs.

A Matlab implementation of the PMC algorithm can be downloaded from http://www.cwi.nl/~sbohte/code/pmc.

2 Cascading MiniMax Classification

The nonparametric Polynomial Cascade Regression Algorithm [6] is based on the premise that very high dimensional nonlinear regression can be done using a finite number of low dimensional structural units, which are added one at a time to the regression function. By keeping the structural units low dimensional, the algorithm is able to produce stable, accurate, regression functions in very high dimensional, large problem domains (i.e. with tens of thousands of features and tens of thousands of training examples [6, 7]). These regression models have excellent performance, both in terms of regression accuracy and scaling: the algorithms scale linearly with the dimensionality of the problem-space, and linearly with the number of examples.

However, the Polynomial Cascade Regression Algorithm is not suitable for classification because it optimizes an error metric that typically doesn’t create an effective classification model. Mainly, Polynomial Cascade Regression minimizes least squared error, treating classification as regression by fitting a continuous regression surface to class labels (for example, -1 and +1 for binary classification). In contrast, algorithms that build effective classifiers, such as boosting [4], support vector machines [1], and MPMC [2, 3], fit to metrics that only attempt to separate classes.

In this section, we describe an adaptation of the Polynomial Cascading algorithm to nonparametric binary classification using the MPMC framework.
**Problem Definition**  Let \( x \) and \( y \) denote the set of training samples available in a binary classification problem, with \( x, y \in \mathbb{R}^{d \times N} \), for \( N \) samples, each of dimensionality \( d \). The means and covariance matrices are denoted respectively by \((\bar{x}, \Sigma_x)\) and \((\bar{y}, \Sigma_y)\). Let \( x_i \) and \( y_i \) denote the respective vectors in dimension \( i \), \( \{i = 1 \ldots d\} \). The problem is to construct a classifier that efficiently and accurately separates unseen data from the same respective classes.

**MPMC**  The Minimax Probability Machine Classification algorithm in [3] determines a hyperplane \( H(a, b) = \{z|a^Tz = b\} \), where \( z, a \in \mathbb{R}^m, b \in \mathbb{R} \) (for some dimension \( m \)), which separates two classes of points, \( u \) and \( v \), with maximal probability with respect to all distributions having these means and covariance matrices:

\[
\max_{a \neq 0, b} \alpha \quad \text{s.t.} \quad \inf_{u \sim (\mu, \Sigma_u)} \Pr\{a^T u \geq b\} \geq \alpha \\
\inf_{v \sim (\nu, \Sigma_v)} \Pr\{a^T v \leq b\} \geq \alpha,
\]

the value \( 1 - \alpha \) then denotes the estimate of the maximum misclassification probability bound, and the MPMC algorithm of [3] minimizes this bound.

### 2.1 Polynomial MPMC Cascade

The general idea behind the Polynomial MPMC Cascade algorithm is to start off with a low dimensional structure for the first cascade level: this structure is derived from a polynomial of just one input dimension (attribute) of the data vectors, where the particular input dimension is selected from all \( d \) input dimensions by computing the class separation power (i.e. the \( 1 - \alpha \) error rate in (1)) of the corresponding polynomial with MPMC. Then, the next level is constructed by combining the output of this structure with a new input dimension, where again this input dimension is selected by trying all \( d \) input dimensions, i.e.: take dimension \( i = (1 \ldots d) \), create a polynomial of both the output from the previous level and the vector of input dimension \( i \), and determine the usefulness of this polynomial structure for separating the classes with MPMC. Then, the best separating polynomial structure is selected as an additional level to the cascade. The output of this level is a weighted sum of the output of the previous level and the new polynomial: we use MPMC to determine this weighting, thus obtaining a (decreasing) classification error bound at every level as we construct the cascade. We keep adding levels until this classification error bound no longer improves. The procedure is depicted in Figure 1a.

Formally, the procedure works as follows:

First the set of training samples \( z = x \cup y \) is linearly scaled, that is, for each input dimension the maximal and minimal value of \( z_i \in \mathbb{R}^N \) are determined (\( z_i \) the vector of values in the training samples for input dimension \( i \)), and \( z_i \) is linearly scaled to the range \([-1,1]\) with scaling vectors \( c_0, d_0 \in \mathbb{R}^d \) (slope, intercept).

To build the first cascade level, we define a second order candidate polynomial \( Z^i_0 \), for each input dimension \( i = (1 \ldots d) \), as:

\[
Z^i_0 = (z_i, z^2_i),
\]

where \( z_i = x_i \cup y_i \); \( Z^i_0^+ \) and \( Z^i_0^- \) denote the parts of \( Z^i_0 \) from the respective classes. For each candidate input dimension \( i \), we compute the means and covariance matrices of \( Z^i_0^+ \) and \( Z^i_0^- \) : \( (\bar{Z}_{0i}^+, \Sigma_{Z_{0i}^+}) \) and \( (\bar{Z}_{0i}^-, \Sigma_{Z_{0i}^-}) \). Plugging these values into MPMC, we obtain hyperplane coefficients \( a_i, b_i \) and error bound \( s_i \). We select that input dimension that has the minimal error bound: \( S_1 = \min(s_i) \), where \( S_1 \) is the error bound for the first level.
MiniMax is constructed of the form:

$$Z = (a, b, s)$$

As before, we then use MPMC to find the dimension and with input $$G_{j-1}$$, the output vector is constructed of the form:

$$Z_j = (z_i, z_i^2, z_i G_{j-1}, G_{j-1}, G_{j-1}^2)$$.

As before, we then use MPMC to find the dimension $$i$$ associated with the polynomial $$Z_j$$ with the minimal classification error bound. We compute the linearly scaled output vector of the level $$j$$ as

$$G_j = c_j^1 [a_i^T z_i^0 - b_i] - d_i$$.

Crucially, the output $$G_j$$ of cascade level $$j$$ is then computed as a weighted sum of the output $$G_{j-1}$$ from level $$j - 1$$, and the output of the new structure $$G_j$$. The weighting is computed using MPMC on $$G_{j-1}$$ and $$G_j$$. Define:

$$g_j^+ = (G_{j-1}^+, G_j^+)$$
$$g_j^- = (G_{j-1}^-, G_j^-)$$

then we compute the MPMC of $$(g_j^+, \Sigma_{g_j^+})$$ and $$(g_j^-, \Sigma_{g_j^-})$$. MPMC yields combination coefficients $$\beta_1 \in \mathbb{R}^2, \gamma_j \in \mathbb{R}$$, and error bound $$S_j$$, where $$S_j$$ is the classification error bound for level $$j$$.

The properly scaled output vector of the level $$j$$, $$G_j$$ is then computed as $$G_j = A_j [\beta_j G_{j-1}^+ + \beta_j G_{j-1}^- - \gamma_j] - B_j$$, $$\beta_1$$ denotes the first element of $$\beta_j$$, $$\beta_2$$ the second, $$A_j$$ and $$B_j$$ are the linear scaling factors. The construction of new levels stops when the classification error bound $$S_j$$ no longer decreases ($$S_{j-1} - S_j \leq \epsilon$$).
Improving error bound  While constructing the Polynomial Minimax Cascade (PMC), the error-bound on the classification, $S_i$, monotonically decreases because as another level $j$ is added, the MPMC attempts to find the best classification given previous cascade output $G_{i-1}$, and the new structure’s output $g_j$. At worst this classification will be as good as that obtained in $G_{i-1}$ (which is our stopping criteria), and if there is any additional discriminatory information contained in $g_j$, the error-bound will be better, i.e. decrease. The error-bound $S_L$, with $L$ the final level, is our estimate for the maximum error bound on the test set.

Evaluating the cascade  The cascade is evaluated using a set of unseen test samples: $u \cup v$, $u$ and $v$ from the respective binary classes. This set is first linearly scaled with $c_0, d_0$, after which all values outside the range [-1,1] are clipped to [-1,1]. At every level, we take the input dimension selected in the construction of the cascade, and compute the (scaled and clipped) polynomial that is the output of the level, including the weighting of the level’s structure with the (clipped) output of the previous level: $G_j = A_j [a_1 g_{j-1} + a_2 f_j - b] - B_j$. The output values of a cascade’s level are subsequently used as input for the next level, together with the additional input dimension of that level (see also figure 1b). After thus running the set of test samples through all levels, we obtain an output classification vector with range [-1,1], discriminating the test set $u \cup v$ into two classes at 0.

Complexity  It is easy to see that the complexity of the algorithm is linear in the number of samples $N$, the number of dimensions of the input $d$, and the number of levels $L$: $c^3 \times N \times d \times L$, where $c^3$ is a constant related to computing the MPMC, with $c$ being the order of the polynomial (for our $Z_j = (z_i, z_i^2, z_i G_{j-1}, G_{j-1}, G_{j-1}^2)$, the value for $c$ is 5) [3].

Projecting onto data-space  Intuitively, the use of only the input-dimensions as potential building blocks for the low-dimensional structures of the cascade seems limiting. We propose a variation of PMC where instead of the actual input-dimensions, we use as “input dimensions” projections of the training-data onto single training samples: $z_i = z^T_i z / d$, where $i = (1 \ldots N)$, $z_i$ is the thus constructed “input dimension”, $z_i^T$ is training example $i$, and $z$ is the set of all training examples. As in the PMC explained above, polynomials of every $z_i$, $i = (1 \ldots N)$ are evaluated, and the one polynomial most effectively separating the classes is added to the cascade using MPMC (as above).

This projection procedure increases the number of available building blocks for the cascade, albeit at the cost of speed as the number of examples is typically much larger than the number of input dimensions. Although the datasets explored here are small enough for this not to be a problem, for very large datasets it might be useful to take a random sample from the data, instead of trying all $N$ data vectors.

Nonparametric  The one parametric choice we make in the PMC is the complexity of the polynomial: here, we chose a simple quadratic polynomial. More complex polynomials can be chosen, but may increase the risk of overfitting the training samples.

3 Results  We studied the performance of the PMC algorithm for a number of benchmark problems, mostly the benchmarks used in [3]: Wisconsin breast cancer dataset, Pima diabetes, Ionosphere and Sonar data (as obtained from the UCI repository). Additionally, we tested on the House-voting dataset. As in [3], each dataset was randomly partitioned into 90% training and 10% test sets. The data for the Twonorm problem was generated as specified by Breiman [8]. The results we report in Table 1 are the averages over 100 random partitions.
We studied this issue by tracking the performance of the polynomial in the cascade, we attempt to minimize the possibility of overfitting demonstrated in [3], this confirms the notion that our nonparametric MPMC-based algorithms.

Polynomial Cascade algorithms.

Learning and overfitting: In the class of PMC algorithms, we have one free parameter: the order of the polynomial structure. By using the minimal - quadratic polynomial in the cascade, we attempt to minimize the possibility of overfitting the training-samples. We studied this issue by tracking the performance of the

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PMC Dim</th>
<th>PMC Data</th>
<th>PMC Mix</th>
<th>Lin MPMC</th>
<th>Gauss MPMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twonorm</td>
<td>92.2 ± 0.1</td>
<td>96.8 ± 0.1</td>
<td>96.5 ± 0.1</td>
<td>96.8 ± 0.4</td>
<td>95.7 ± 0.5</td>
</tr>
<tr>
<td>(α)</td>
<td>(96.2 ± 0.2)</td>
<td>(96.2 ± 0.2)</td>
<td>(97.6 ± 0.2)</td>
<td>(84.4 ± 0.1)</td>
<td>(91.3 ± 0.1)</td>
</tr>
<tr>
<td>Cancer</td>
<td>96.8 ± 0.2</td>
<td>97.1 ± 0.2</td>
<td>96.7 ± 0.2</td>
<td>97.0 ± 0.4</td>
<td>96.8 ± 0.3</td>
</tr>
<tr>
<td>(α)</td>
<td>(95.8 ± 0.1)</td>
<td>(95.6 ± 0.1)</td>
<td>(96.2 ± 0.1)</td>
<td>(84.4 ± 0.1)</td>
<td>(89.1 ± 0.1)</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>91.4 ± 0.4</td>
<td>89.6 ± 0.5</td>
<td>90.9 ± 0.5</td>
<td>83.4 ± 0.9</td>
<td>91.5 ± 0.7</td>
</tr>
<tr>
<td>(α)</td>
<td>(91.3 ± 0.2)</td>
<td>(85.4 ± 0.3)</td>
<td>(92.5 ± 0.2)</td>
<td>(65.5 ± 0.3)</td>
<td>(89.3 ± 0.2)</td>
</tr>
<tr>
<td>Diabetes</td>
<td>76.2 ± 0.5</td>
<td>74.4 ± 0.5</td>
<td>75.9 ± 0.5</td>
<td>76.3 ± 0.6</td>
<td>76.2 ± 0.6</td>
</tr>
<tr>
<td>(α)</td>
<td>(38.2 ± 0.1)</td>
<td>(33.8 ± 0.1)</td>
<td>(38.2 ± 0.1)</td>
<td>(32.2 ± 0.2)</td>
<td>(32.5 ± 0.2)</td>
</tr>
<tr>
<td>Sonar</td>
<td>81.7 ± 0.7</td>
<td>84.8 ± 0.9</td>
<td>81.2 ± 0.8</td>
<td>74.9 ± 1.4</td>
<td>87.5 ± 0.9</td>
</tr>
<tr>
<td>(α)</td>
<td>(95.7 ± 0.2)</td>
<td>(93.5 ± 0.3)</td>
<td>(96.1 ± 0.2)</td>
<td>(67.0 ± 0.4)</td>
<td>(99.9 ± 0.1)</td>
</tr>
<tr>
<td>Voting</td>
<td>94.8 ± 0.3</td>
<td>94.8 ± 0.3</td>
<td>95.1 ± 0.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(α)</td>
<td>(93.0 ± 0.2)</td>
<td>(94.8 ± 0.2)</td>
<td>(97.5 ± 0.1)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Performance of PCM, for the PMC Dim, PMC Data and PMC Mixed variants. Reported are test-set accuracy (TSA) and on the next line the lower error bound α (all percentages). Results compared to those reported for linear and Gaussian kernel based MPCM. Note that for PMC Dim, the Twonorm accuracy increases to 94.2/94.6 % with larger training sets (600 resp 900). In Votes, missing attributes were replaced by 0.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PMC Dim</th>
<th>PMC Data</th>
<th>PMC Mix</th>
<th>SVML</th>
<th>SVMG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twonorm</td>
<td>92.2</td>
<td>96.8</td>
<td>96.5</td>
<td>95.1</td>
<td>96.1</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>95.8</td>
<td>97.1</td>
<td>96.7</td>
<td>96.4</td>
<td>96.5</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>91.4</td>
<td>89.6</td>
<td>90.9</td>
<td>87.1</td>
<td>94.1</td>
</tr>
<tr>
<td>Pima diabetes</td>
<td>76.2</td>
<td>74.4</td>
<td>75.9</td>
<td>77.9</td>
<td>77.9</td>
</tr>
<tr>
<td>Sonar</td>
<td>81.7</td>
<td>84.8</td>
<td>81.2</td>
<td>76.1</td>
<td>86.6</td>
</tr>
</tbody>
</table>

Table 2: Performance of PMC variants compared to Linear SVM (SVML) and Gaussian kernel SVM (SVMG) results (standard deviations omitted per high similarity)
Figure 2: Performance of PMC variants on the data during the construction of the cascades. Shown are the data for Ionosphere, Wisconsin Breast Cancer and Sonar. In each graph is plotted the averages of Learning Set Accuracy (LSA), Test Set Accuracy (TSA) and error-bound, as a function of the number of levels in the cascade during construction.

algorithm variants during the construction of the cascades (for all 100 runs): at every level, we noted the current error-bound and we computed the accuracy on the training and the test set. The results for three benchmarks for all algorithm variants are shown in figure 2. Shown are the averages over 100 runs, where the values for those cascades that are completed (met stopping criteria) are taken as constant for computing performance for levels larger than the size of these cascades.

The graphs clearly show that the performance of the cascades on the test samples is practically constant after initial learning (also observed in the other benchmarks, not shown). Although some benchmarks show slightly better performance early on, this seems to be within the variance of the final results.

4 Conclusion

The Polynomial MPMC Cascade (PMC) class of classifiers introduced in this paper demonstrates excellent performance in the key areas that determine the usability of a classifier: accuracy, speed, scalability to high dimensional problems, and a minimum of “tinkering” learning parameters. As we have shown, the proposed class of algorithms is essentially nonparametric and highly accurate on the presented benchmarks, and computationally it is linear in complexity in the dimension of the problem, in the number of training examples, and in the size of the cascade.
Although all three versions of the PMC framework studied here demonstrated good error rates on test data, the bounds for the version that used both input dimensions and data-projections as cascade building blocks (PMC Mixed, see Results section), tended to be overoptimistic for some error bound predictions. Since the MPMC framework requires estimates of mean and covariance matrix, inaccuracies in these estimates lead to inaccuracies in error bounds. One solution to this problem we are currently investigating is to attempt to determine when these estimates are poor, and to compensate for this using a method similar to the Robust MPMC framework defined in [3] or the robust MPM regression framework defined in [9, 10].

We can see several areas where the proposed class of algorithms could be extended: for instance by using higher-dimensional (3,4) structures as building blocks, and, as noted above, the robust estimates of error bounds to automatically and efficiently choose the best model. In addition, we find fast asymptotic convergence of the test set accuracy as the cascade is constructed in all the datasets tested (i.e. fig 2). This suggests that the number of levels in the cascades could be reduced, creating more compact (sparse) models. We are currently investigating the use of robust error bounds to attempt to identify when further addition of cascade structure will not lead to significant improvement in test set accuracy.

In conclusion, we find that the proposed class of Polynomial MPCM Cascade Classifier algorithms offer a “Plug & Play” solution for supervised classification problems, and warrant further study. A Matlab implementation of the PMC algorithm can be downloaded from http://www.cwi.nl/~sbohte/code/pmc.

Acknowledgement. We thank Gert Lanckriet for making publicly available his MPMC implementation. Work of SMB supported by the Netherlands Organization for Scientific Research (NWO), TALENT grant S-62 588.

References