# Linear Time Nonparametric Classification and Feature Selection with Polynomial MPMC Cascades for large datasets

Markus Breitenbach[1], Sander M. Bohte[1,2], and Gregory Z. Grudic[1]

[1] University of Colorado at Boulder, Boulder CO 80309, USA,
{grudic},{breitenm}@cs.colorado.edu
[2] CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
sbohte@cwi.nl

**Abstract.** The recently proposed Polynomial MPMC Cascade (PMC) algorithm is a nonparametric classifier for high-dimensional non-linear binary classification with performance competitive with state-of-the-art classifiers like SVMs. Importantly, the algorithm has linear-time complexity with respect to both training-set size and dimensionality of the problem. In this paper, we show how we can exploit this computational efficiency to build classifiers from very large datasets typical in datamining problems. Furthermore, we demonstrate that the PMC algorithm efficiently does feature selection in such very large large problem domains. Experimental results are given on datasets ranging in sizes between 170,000 and 4.8 million examples. We empirically verify the linear time dependence of the algorithm, and explore how the stability of the classifier is influenced by sample size. The techniques discussed in this paper should allow nonlinear binary classifiers to be efficiently learned from tens of millions of training data, with feature selection being a added byproduct.

## 1 Introduction

An important problem in machine learning, and data mining in particular, is the analysis of truly large datasets. Extracting relevant structures from datasets that range from hundreds of thousands of examples to many millions clearly requires a highly efficient method for doing so, and most state-of-the-art machine learning techniques are computationally too expensive to use on the entire set (e.g. Support Vector Machines are cubed in the number of data-points).

To use such methods, inevitably one has to (cleverly) reduce the data to a manageable size, be it by taking a sample of the examples, and/or by trying to select only the most relevant features. E.g. ASVM[1] is a reformulation of the linear Support Vector Machines optimization problem where the algorithm finds a solution to the SVM problem using only a smaller "active set" during the learning process and by solving a number of linear equations (no inequalities). In [2] Yu *et al.* suggest using clustering to generate a smaller training set for SVMs to learn on. This method requires multiple runs of small SVM problems, but does

not support the use of kernels yet, i.e. it can only handle linearly separable cases. Domingo and Watanabe [3] suggest a scalable version of boosting, however, this classifier can also only learn linear decision surfaces due to it's limitation to 1-level tree stubs.

Other methods include building several models with subsets of the data [4]. However, these methods may not be able to guarantee the accuracy of the final result to be as good as an individual learning algorithm applied to the entire data set, since a considerable amount of information may not be accessible to each of the separate learning processes.

To really use all the available examples in very large dataset, a suitable machine learning method clearly has to have as low time complexity as possible.

Here, we examine the Polynomial MPMC Cascade (PMC) algorithm [5], a recently proposed nonparametric classifier for non-linear binary classification. The PMC constructs classification models in linear time and has been shown to have competitive performance compared to other state-of-the-art classifiers like Support Vector Machines (SVM) or the Minimax Probability Machine for classification (MPMC) [6]. We show that the PMC can be used to generate (very competitive) classifiers for very large datasets while using all the available data.

The PMC classifier is based on cascaded low dimensional polynomial structures, where each separate level of the cascade is constructed using the Minimax Probability Machine Classifier (MPMC) algorithm. The PMC algorithm learns the classification function in linear time with respect to both training-set size and dimensionality of the problem. The run-time complexity of $O(L \cdot n \cdot d \cdot c^3)$, with $L$ being the number of levels build, $n$ being the number of examples, $d$ being the number of features and $c^3$ being the run-time complexity of one run of the MPMC. In general, computing the MPMC is cubed in the dimensionality of the problem; However, by using cascades of low dimensional polynomials (5D) the computational complexity of the MPMC reduced to a fixed cost ($c^3 = 25$). The number of levels depends on the data, i.e. how many levels are needed to accurately separate the data. Additionally the PMC does not have parameters that need to be fine-tuned to generate a good model. This avoids the lengthy process of determining a good set of parameters using cross validation. For example, in Support Vector Machines (SVMs) [7], both the choice of kernels and corresponding kernel parameters is based on extensive cross validation experiments. Other algorithms, such as kernel versions of the Minimax Probability Machine Classification (MPMC) [8], Neural Networks and ensemble methods like Boosting [9], suffer from the same computational pitfalls.

In this paper, we demonstrate that the linear time complexity of the PMC algorithm makes it suitable for accurately classifying very large datasets. As per the law of large numbers, with any classifier there should be a point where adding more training examples should not make a significant change in classification performance on the test-set. Since the PMC algorithm has linear time-complexity, it allows us to examine the classification performance when using increasingly larger sample sizes of the training data, thus giving empirical insight into the question "how much is enough?". We compare these results to predictions from

recent (theoretical) sampling method. We present experiments with the $\mathcal{AA}$ sequential sampling algorithm [10] to determine the (minimal) number of samples needed to arrive at an effective model.

In addition to these results, the speed of the model construction suggests new ways of determining which features of the dataset are important and use this to speed up the PMC model construction. We show that the most important features can be determined quickly by building a set of PMC models using small sample sizes. When constructing PMC models using the reduced dataset containing only these relevant features, we retain the same classification performance as with the complete dataset.

This paper is organized as follows: in Section 2 we introduce the PMC algorithm. In section 3 we show the performance of the PMC algorithm for a number of very large datasets, and for a range of sample sizes. Additionally, we empirically show that the algorithm is indeed a linear-time algorithm. In Section 3.1 we explore a sequential sampling methodology to predict a sufficiently large sample size. In Section 4, we show how we use the PMC algorithm to quickly perform feature selection. In Section 5 we discuss the experimental evidence and possible future work.

The code used in the experiments is available at
`http://ucsu.colorado.edu/~breitenm/pmc.html`.

## 2 The Polynomial Minimax Cascade

In this section, we describe the Polynomial MPMC Cascade algorithm for nonparametric binary classification as introduced in [5].

The general idea of the Polynomial MPMC Cascade (PMC) algorithm is that a very high dimensional nonlinear classifier can be constructed using a finite number of low dimensional structural units that are successively cascaded one at a time to the classification function. The PMC is an adaptation of the Polynomial Cascade Regression Algorithm [11] adapted to nonparametric binary classification using the MPMC framework [8]. Applying the MPMC algorithm to any binary classification task yields the parameters for a (Minimax) optimal hypersurface, and an estimate for the maximum classification error bound. In the PMC, MPMC is used to select the best polynomial structure at each level in a greedy fashion by taking the polynomial structure that has the lowest classification error bound (as determined by the MPMC). Subsequently MPMC is used again to combine the previous level's output with the output of the new structural unit.

In a broad outline: the cascade starts 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 of the corresponding polynomial with MPMC. Then, the next level is constructed by combining the output of the previous structure with a new input dimension, where again this input dimension is selected by trying all $d$ input di-

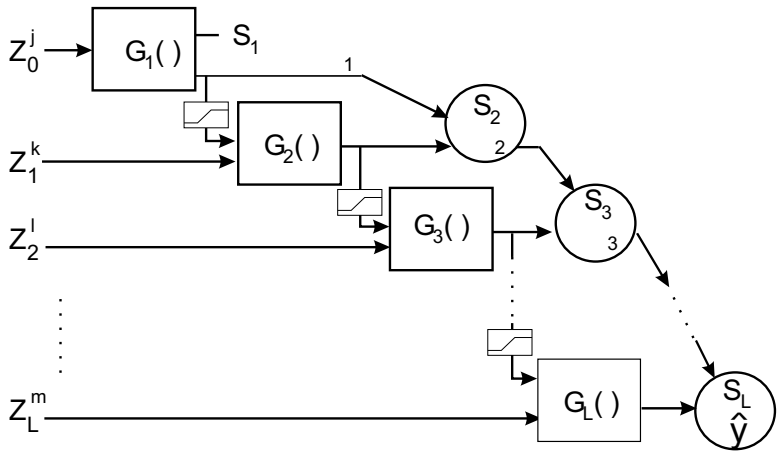


**Fig. 1.** Polynomial Minimax Cascading Classifier construction: in each level the input is determined for which the low dimensional polynomial is best separated by the MPMC procedure, yielding the output $G_i$ for level $i$. This output, together with the output of the previous level is put through another MPMC (circle), which optimally 'adds' two levels together (and yields the classification output $M_i$ and the error-bound $S_i$ that is used as a stopping criteria).

mensions, i.e.: take dimension $i = (1 \ldots d)$, create a polynomial of both the input 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 classification output of this level is a weighted sum of the output of the previous level and the new polynomial: MPMC is used to determine this weighting, thus at the same time obtaining a (decreasing) classification error bound $S_i$ at every level as we construct the cascade. We keep adding levels until this classification error bound $S_i$ no longer improves. The procedure is depicted in Figure 1. For a more elaborate definition see [5].

Note that the PMC algorithm is similar to Boosting [9] by using weak classifiers at each level and summing their classification up in a weighted sum. Each level, however, uses the weighted output of the previous level as a feature, instead of re-weighing examples.

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 [8]. The order of the polynomial is the one parametric choice available in the PMC, we use the default quadratic polynomial (thus $c = 5$).

## 3 Classifying large datasets

As noted, no results are known on applying state-of-the-art nonlinear classification algorithms to truly very large datasets. Adaptations of SVM and Boosting

are limited to linear classification surfaces, and methods that split the data into small chunks and then try to combine the many resulting classifiers can potentially miss structure that is only apparent when considering the dataset at large [4]. In principle however, the law of the large numbers suggests that classification performance should level off once the amount of training examples sufficiently determines the underlying distributions, i.e. the expected benefit of having $10 \cdot X$ vs. $X$ training examples should diminish rapidly for increasing $X$.

In this section, we use the PMC algorithm as a state-of-the-art classifier that is fast enough to empirically explore what $X$ is for a number of very large datasets. For each dataset, we compare the PMC classification performance at different sizes of sampled subsets from the entire distribution, and compare that against the performance when using the entire dataset (or as much as fits into memory) [3].

The PMC uses the linear MiniMax Probability Machine (MPMC) at each level to determine the coefficients of the polynomial. The MPMC estimates the mean and the covariance matrix of each class to determine a hyperplane that separates the two classes minimizing the risk of misclassification, and MPMC performance depends only on an accurate estimate of the mean and the covariance matrix. Improvement in the PMC performance from adding more training examples is thus only expected as long as the extra data significantly help improve the mean and covariance estimates.

To examine how many training-examples are needed before the classification improvement becomes asymptotically small, we present classification performance for increasingly large sample sizes used for the construction of each level in the PMC cascade. We show experiments for a number of very large datasets.

*KDD Cup 1999.* The KDDCUP 1999 dataset is a very large dataset, with 5.1 million examples and includes a wide variety of network intrusions simulated in a military network environment.

The task is to construct an intrusion detector, a predictive model capable of distinguishing between various attacks like probes or privilege elevation and *good* normal sessions. To create a binary classification, we only distinguish between *good* and *bad* examples.

The training set contains 4.8 million examples, and the test set contains 311.000 examples. Nominal attributes in the data are converted to binary attributes by assigning one extra dimension per possible attribute. Additionally, we summarized attributes that were indicative for only one class in the training set into a single attribute. The final data set had 75 features. Note that the test data is not from the same probability distribution as the training data: to make the task more realistic specific attack types not in the training data were included.

[3] For the experiments we used a Matlab implementation of the PMC algorithm. All experiments were conducted on a Pentium 4, 1.7 MhZ, 1 GB Ram machine using Matlab 6 R13. Note that the code was not explicitly optimized for speed, an optimized C implementation should result in a significant speed increase.

In this case the training data does not fit into memory when loaded with Matlab and we report only on sample sizes increasing to the number of examples that just fit into memory (as can be seen from the resulting graphs, this proved to be quite enough).

*Forest Cover.* Forest Cover is a dataset with over half a million examples, divided over seven classes each denoting a type of tree, and the task is to predict the forest cover type from cartographic variables only. In the original task, the first 11340 examples were used as the training-set, the next 3780 as the validation set, and the remaining 565892 examples were used for testing. We find that the training-set seems to be an optimized subset of the examples, as the class distributions in the training set do not reflect the true class distributions: when we construct a PMC model from a randomly selected (equally large) subset from the test-set, the model performs substantially worse than one constructed from the training-set (figure 2f). On the original task, the PMC algorithm had an error rate of 35% versus the published 30% error rate of a neural network. Note, however, that the 35% error rate was reached by just running the algorithm; no effort was spend to improve upon this result.

In all other forest-cover results presented, we removed the first 11340 examples and use the original testing set in a 90/10 split for training and testing. As classes $4-7$ are only a very small fraction of this set, we only considered classes $1-3$ as binary classification tasks.

*ICML 2004 Physiological Data set.* This dataset was published as part of the ICML 2004 workshop on modeling physiological data. Three different binary classification tasks were made available, with the additional complexity that two of the tasks contain examples that can be part of both classes. For our experiments we removed the data that can be part of both classes, because they skewed the distributions too much. We split the data randomly 90%/10% into a training and a test set. This leaves us with a training set of 580000 training examples for the first task and 170000 for the second and third task.

**Asymptotic Classification Performance.** To determine how classification accuracy changes with increasing sample size, we build 10 models for each possible sample size. Plotted in figure 2: the sample size against the error-rate and standard deviation of the mean on the test-set of the various models. We see that for all five datasets, the error-rate stabilizes after a certain sample size, and the standard deviation decreases asymptotically decrease.

For the KDD'99 dataset for example, the classifier accuracy does not improve anymore when using more than 60000 samples. Note that the PMC accuracy is 0.92 percent, equally good the winning entry into the KDD 1999 competition [12] with 0.927 percent (for the same binary classification task). Note that our total runtime was around 2 hours using unoptimized Matlab code vs. one day processing time on a 2 processor Ultra SPARC. For Forest Cover and all the Physiological tasks we can see the error stabilize when using more than 100000

samples. For Forest Cover and the Physiological tasks we are not aware of published results regarding runtime or error rates in the way we setup our experiments. The error rates for Forest Cover Class $1-3$ were 0.2322, 0.2789 and 0.1873 respectively. The error rate for the Physiological data sets were 0.012 for task 1, 0.32 for task 2 and 0.083 for task 3.

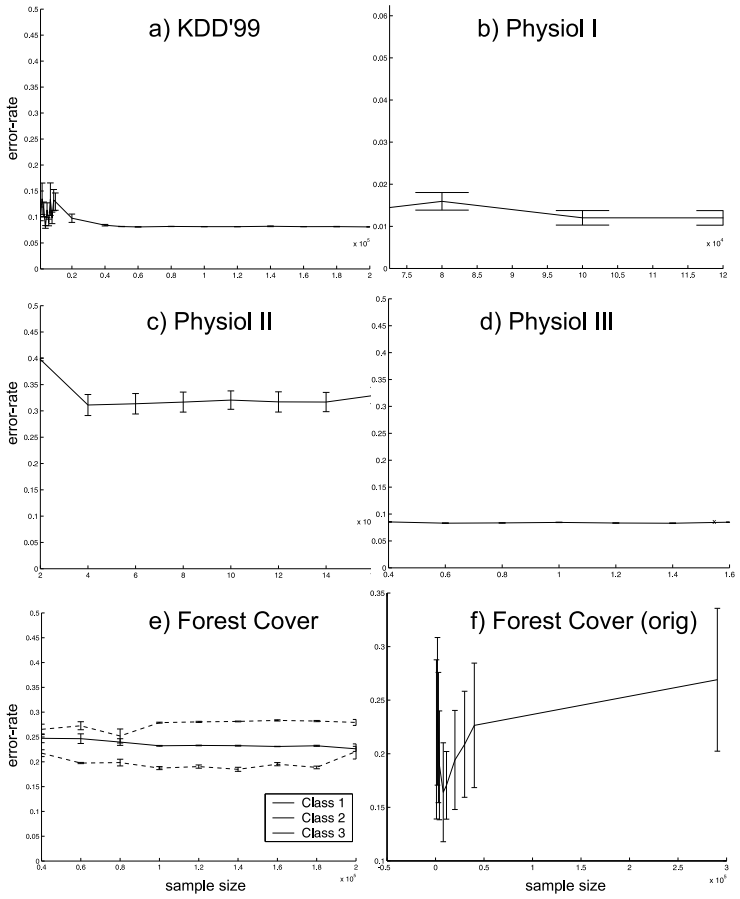


**Fig. 2.** a-e) error rate converges asymptotically for larger sample size. f) As described in the text, the Forest Cover data set has an anomaly that causes the error rate to go up if more than the specially crafted 11340 training examples are being used.

**Linear Time Model Construction** In figure 3, we plotted the time required for computing the models versus the sample size. The graph shows that the computational complexity is indeed roughly linear in the size of the training set. This implicitly shows that the number of levels (the length of the cascade),

which is data dependant due to the stopping heuristic being used, does not change significantly as the number of examples increase.

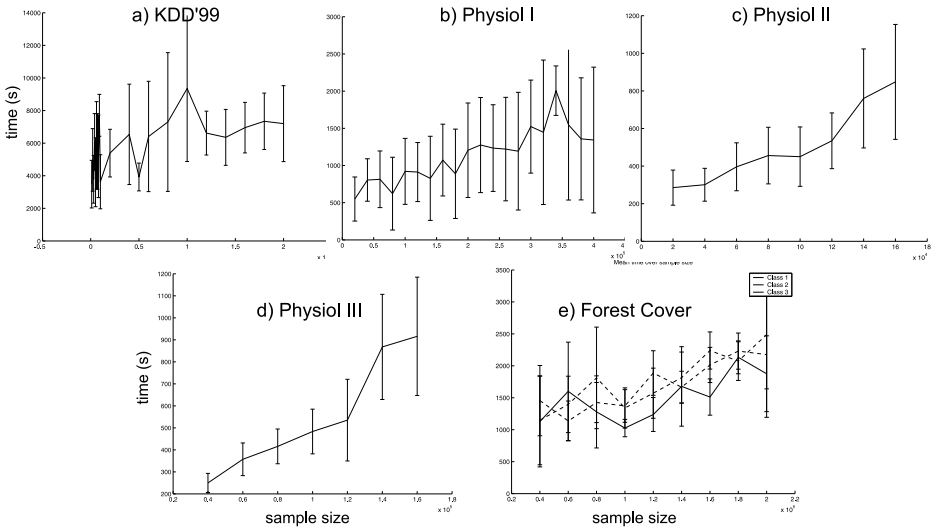


**Fig. 3.** Model construction time is linear in the number of examples.

The time complexity of the PMC is also linear in the number of levels, or the length of the cascade. Since the construction of the cascade is data dependent, we checked how much the number of levels changes when larger sample sizes are used. We found that the number of levels remains the same or increases only slightly for larger sample sizes.

Note the user can abort the learning process of the PMC at any point in time and use the hypothesis that has been learned up to that point in time. This is due to the fact that the PMC algorithm always improves upon the previous levels hypothesis. The hypothesis of most other algorithms can be arbitrarily worse, if the stopping criteria has not been reached. This fact makes the PMC algorithm suitable for "real-time" tasks that require a hypothesis within a certain amount of time. After the second level it is possible to give a rough estimate of how many levels can be build within a certain amount of time.

### 3.1 Theoretic Sequential Sampling

The PMC algorithm uses the MPMC as a "weak" classifier to build each level. At each level, a five dimensional problem, determining coefficients for the polynomial that minimizes the risk of misclassification, is solved by the classifier. The solution of the MPMC's optimization problem depends solely on the mean and covariance matrix for each class of the data and the training data is solely used

for the purpose of determining an estimate of the mean and the covariance matrix. An appealing idea is to sample from the training data until some stopping criteria tells us that we've seen enough data to compute mean and covariance within some factor $1 + \epsilon$.

The notion that there is a limit to the number of data-points needed to estimate some variable up to a certain precision has been treated in a number of papers on sequential sampling. The $\mathcal{AA}$ algorithm [10] is a provably optimal algorithm for Monte Carlo estimation. It uses sequential sampling to estimate the mean $\mu$ of the data by running experiments repeatedly that produce a random variable $Z$ (i.i.d.) in $[0, 1]$ with $E[Z] = \mu$. The algorithm, given $\epsilon$ and $\delta$, produces an estimate that is within a factor of $1 + \epsilon$ of $\mu$ with a probability of at least $1 - \delta$ running only the minimum number of experiments (optimality) within some constant factor.

This estimate of the required number of samples needed for a set precision should be a good indicator of how to choose a sample size for the PMC that is sufficiently large, since as noted, the PMC algorithm depends on the estimates of the mean and covariance.

We use the $\mathcal{AA}$ algorithm to predict how many examples we need for an estimate of the mean that is within the bounds of $\epsilon$ with a high probability. We make the assumption that, if the mean is "good enough" for all of the features, then we can assume the quality of the estimate of the covariance matrix is fairly similar. The predictions for the different datasets and different values of $\epsilon$ are given in table 1.

As can be gleaned from table 1, the main issue is how to choose a reasonable value. Comparing the predicted required sample sizes with figure 3 shows that only predictions with $\epsilon \leq 0.01$ give estimates that are not too small, i.e. an estimate for the sample size that is sufficient would yield a classifier that can make accurate predictions. Unfortunately, for $\epsilon \leq 0.01$ the sample-size estimates are very conservative and predict a number of samples for the mean that is usually far larger than the actual, experimentally determined minimum number of samples at each level.

Note that we have no theoretical motivation for setting $\epsilon$ to this value, and in a sense we have traded one unknown parameter for another. The only benefit from using $\epsilon$ is that it can be considered a more interpretable parameter.

## 4 Subset sampling for Feature Selection

The linear time complexity of the PMC algorithm opens new venues for selecting (and using only) relevant features. In this Section, we propose one such feature selection method.

The asymptotically improving classification accuracy with larger sample sizes of the PMC models as shown in Section 3 can arise from two sources: 1) at each level, the MPMC can determine increasingly optimal decision boundaries, and 2) the PMC finds new features (or orderings of features) that help improve the classification. We hypothesize that the main improvement stems from the

| | $\epsilon = 0.35$ | $\epsilon = 0.05$ | $\epsilon = 0.01$ | $\epsilon = 0.001$ |
|---|---|---|---|---|
| Forest Class 1 | $4700 \pm 300$ | $16000 \pm 900$ | $3.0 \cdot 10^6 \pm 7.7 \cdot 10^3$ | $2.5 \cdot 10^7 \pm 1.4 \cdot 10^5$ |
| Forest Class 2 | $4800 \pm 300$ | $16000 \pm 1000$ | $2.9 \cdot 10^6 \pm 6.6 \cdot 10^3$ | $2.4 \cdot 10^7 \pm 2.9 \cdot 10^5$ |
| Forest Class 3 | $4800 \pm 400$ | $1600 \pm 900$ | $2.9 \cdot 10^6 \pm 1.1 \cdot 10^4$ | $2.4 \cdot 10^7 \pm 3.3 \cdot 10^5$ |
| KDD Cup 99 | $3300 \pm 180$ | $9600 \pm 800$ | $1.8 \cdot 10^6 \pm 8.4 \cdot 10^3$ | $1.4 \cdot 10^8 \pm 2.0 \cdot 10^5$ |
| Physiological 1 | $2600 \pm 150$ | $2300 \pm 200$ | $4.0 \cdot 10^5 \pm 1.0 \cdot 10^3$ | $3.4 \cdot 10^6 \pm 4.6 \cdot 10^4$ |
| Physiological 2 | $3300 \pm 100$ | $2100 \pm 50$ | $3.9 \cdot 10^5 \pm 1.0 \cdot 10^3$ | $3.2 \cdot 10^6 \pm 2.9 \cdot 10^4$ |
| Physiological 3 | $3400 \pm 80$ | $2500 \pm 35$ | $4.5 \cdot 10^5 \pm 0.4 \cdot 10^3$ | $3.7 \cdot 10^6 \pm 2.8 \cdot 10^4$ |

**Table 1.** Sample size predictions for different values of $\epsilon$. The probability of failure $\delta$ was set to 0.05.

first source, because the number of levels used in the models does not increase dramatically. Also manual inspection of the models built reveals that in many cases the same features are selected. Here, we examine whether (a number of) PMC models build using small sample sizes can be used to determine a subset of the features from which to build a PMC model that uses large sample size.

In figures 4a-e, we show histograms for the features used in 10 models using a limited sample-size (dark columns) compared to the features used in 10 models using a very large sample size (light columns).

In the upper half of Table 2, we show the error rate for the data sets: Forest Cover for classes $1-3$, Kdd Cup 99, ICML-Physiological I, II and III. We report the average error-rate for models build with a large sample-size, the number of features of the original task and their runtime. In the upper half chose a sample size that the data set was known to be stable, the error rate for this sample size as well as the runtime for this sample size if all features are used. In the lower half of the table, we report the sample-size that was used to determine the best features, the number of features selected, the number of features that the small sample size cascades did not use but that were used in the large-sample cascades, and the average error rate for large sample-size PMC models (as reported in the upper half of the table) constructed from only the selected features, as well as the average runtime from using only the selected number of features. The runtime for the feature selection in the lower half is the total runtime of ten runs with the smaller sample size (Sample Size FS).

We see that the reduced number of features resulted in equally good models. In the case of Forest Cover (Class 2) and the Physiological data set (task 3) we saw an increase in runtime. This is due to the fact that several of the models build more levels. The features that were missed by the cascades with the smaller sample size did not have a significant impact on the performance. This is due to the fact that those features were at the very end of the cascades, i.e. they only marginally improved the performance of the classifier.

## 5 Discussion

This paper experimentally verifies that the Polynomial MPMC Cascade (PMC) algorithm can be used to quickly construct accurate non-linear classifiers for

| | Forest 1 | Forest 2 | Forest 3 | Kdd 99 | Phys. 1 | Phys. 2 | Phys. 3 |
|---|---|---|---|---|---|---|---|
| Number of Features | 54 | 54 | 54 | 75 | 14 | 14 | 14 |
| Sample Size | 180000 | 180000 | 180000 | 60000 | 160000 | 160000 | 160000 |
| Error Rate | 0.23 | 0.28 | 0.19 | 0.08 | 0.01 | 0.323 | 0.08 |
| Runtime (sec.) | 1607 | 1263 | 1698 | 6411 | 1204 | 847 | 915 |
| Sample Size FS | 60000 | 60000 | 60000 | 20000 | 60000 | 60000 | 60000 |
| Features selected | 29 | 29 | 29 | 20 | 7 | 6 | 6 |
| Number of Missed | 6 | 6 | 6 | 2 | 0 | 3 | 3 |
| Error Rate FS | 0.23 | 0.28 | 0.19 | 0.08 | 0.01 | 0.30 | 0.08 |
| Runtime (sec.) | 886 | 1332 | 1518 | 6190 | 939 | 675 | 1294 |
| Runtime FS (sec.) | 11206 | 12573 | 7959 | 58405 | 7333 | 3274 | 3669 |

**Table 2.** Feature selection

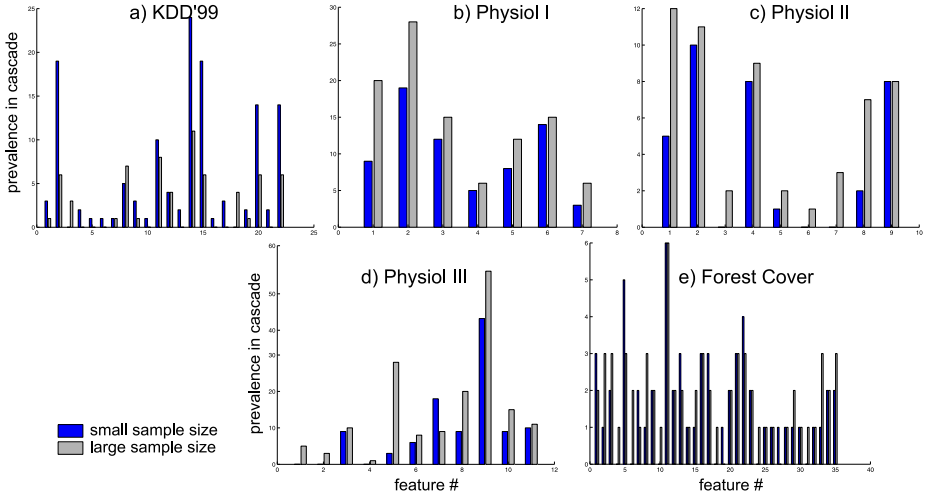


**Fig. 4.** Features selected by 10 PMC models generated with small and large sample sizes.

very large (binary) classification problems. We showed that the computational cost of building a PMC model is linear in sample size, allowing nonlinear binary classifiers to be built from millions of training examples. This differentiates the PMC from other state of the art classifiers, which typically cannot deal with such large problem domains. We further demonstrated that an important added byproduct of a PMC model is automated feature selection.

The classification performance of PMC is asymptotic as sample size increases - i.e. at some point the model effectively stops improving with the addition of more training data. We showed that using the $\mathcal{AA}$ algorithm [10] we can predict a sample-size that is sufficient for a good model. Future work will investigate combining sequential sampling techniques with PMC, such that the algorithm finds the right sample size during the learning process. This type of knowledge

could allow both speed up in classifier learning times, as well as be useful for determining whether enough data exists to build a good model.

*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

1. Mangasarian, O.L., Musicant, D.R.: Active support vector machine classification. (Technical report)
2. Yu, H., Yang, J., Han, J.: Classifying large data sets using svms with hierarchical clusters. In: Proc. ACM SIGKDD. (2003) 306–315
3. Domingo, C., Watanabe, O.: Scaling up a boosting-based learner via adaptive sampling. In: PAC-KDD. (2000) 317–328
4. Chan, P.: An extensible meta-learning approach for scalable and accurate inductive learning (1996)
5. Bohte, S., Breitenbach, M., Grudic, G.: Nonparametric Classification with Polynomial MPMC Cascades. In: Proc. ICML 2004. (2004) to appear, also Technical Report CU–CS–955–03
6. Lanckriet, G.R.G., Ghaoui, L.E., Bhattacharyya, C., Jordan, M.I.: Minimax probability machine. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: Advances in Neural Information Processing Systems 14, MIT Press (2002)
7. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge, MA (2002)
8. Lanckriet, G., Ghaoui, L.E., Bhattacharyya, C., Jordan, M.: A robust minimax approach to classification. Journal of Machine Learning Research **3** (2002) 555–582
9. Freund, Y.: An adaptive version of the boost by majority algorithm. In: Proc. COLT. (1999)
10. Dagum, P., Karp, R., Luby, M., Ross, S.: An optimal algorithm for Monte Carlo estimation. SIAM Journal on Computing **29** (2000) 1484–1496
11. Grudic, G., Lawrence, P.: Is nonparametric learning practical in very high dimensional spaces? In: Proc. IJCAI-97. (1997) 804–809
12. Elkan, C.: (Results of the kdd'99 classifier learning contest; http://www.cs.ucsd.edu/users/elkan/clresults.html)