Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2015, Article ID 216132, 8 pages
http://dx.doi.org/10.1155/2015/216132

Research Article

Cascade Support Vector Machines with Dimensionality Reduction

Oliver Kramer

Computational Intelligence Group, University of Oldenburg, 26111 Oldenburg, Germany

Correspondence should be addressed to Oliver Kramer; oliver.kramer@uni-oldenburg.de

Received 9 November 2014; Revised 13 December 2014; Accepted 28 December 2014

Academic Editor: Sebastian Ventura

Cascade support vector machines have been introduced as extension of classic support vector machines that allow a fast training on large data sets. In this work, we combine cascade support vector machines with dimensionality reduction based preprocessing. The cascade principle allows fast learning based on the division of the training set into subsets and the union of cascade learning results based on support vectors in each cascade level. The combination with dimensionality reduction as preprocessing results in a significant speedup, often without loss of classifier accuracies, while considering the high-dimensional pendants of the low-dimensional support vectors in each new cascade level. We analyze and compare various instantiations of dimensionality reduction preprocessing and cascade SVMs with principal component analysis, locally linear embedding, and isometric mapping. The experimental analysis on various artificial and real-world benchmark problems includes various cascade specific parameters like intermediate training set sizes and dimensionalities.

1. Introduction

Large data sets require the development of machine learning methods that are able to efficiently compute supervised learning solutions. State-of-the-art methods in classification are support vector machines (SVMs) [1]. But due to the cubic runtime and quadratic space of support vector learning with respect to the number of patterns, their applicability is restricted. Cascade machines are machine learning methods that divide the training set into subsets to reduce the computational complexity and employ the principle of dividing a large problem into smaller subproblems that can be solved more efficiently. The cascade principle is outstandingly successful for SVMs [2], as their learning results (the support vectors) can hierarchically be used as training patterns for the following cascade level. The objective of this paper is to show that a further speedup can be achieved via dimensionality reduction (DR) from space $\mathbb{R}^d$ to space $\mathbb{R}^q$ with $q < d$ as preprocessing step without a significant loss of accuracy. In each cascade level of the novel cascade variant, which we will call extreme cascade machines (ECMs), the original patterns (with original pattern dimensionality d) are employed. The support vector learning process on the patterns with reduced dimensionality q turns out to choose a similar set of support vectors like the original SVM learning process on patterns with dimensionality d. The advantage of the proposed combined cascade learning and dimensionality reduction procedure is that the final SVM is also trained on a subset of the patterns with *original* dimensionality d. Hence, an independent test set does not have to be mapped to a low-dimensional space before it can be subject to the ECM classification process.

In this work, we present the approach to divide the training set into subsets, reduce its dimensionality, and employ the cascade principle. The approach turns out to depend on various parameters that are analyzed experimentally. The hybridization of the cascade approach with DR methods belongs to the successful line of research on DR-based preprocessing in supervised learning. At the same time, cascades share similarities with ensemble methods, which combine the learning results of multiple learners and have proven to be strong means to strengthen computational intelligence methods. For example, Ye et al. [3] introduced a k-nearest neighbor based bagging algorithm with pruning for ensemble

SVM classification. Ensembles have also proven well in other applications like visualization of high-dimensional data with neural networks [4].

This paper is structured as follows. In Section 2, the new ECM approach is presented. It is experimentally analyzed in Section 3 with respect to training set sizes, the choice of parameters, and the employment of various DR reduction methods. Conclusions are drawn in Section 4.

2. Extreme Cascade Machines

In this section, we introduce the concept of ECMs. They are based on the combination of classic cascade SVMs with DR preprocessing.

2.1. Support Vector Machines. SVMs for classification place a hyperplane in data space to separate patterns of different classes. Given a set of N observed patterns $\mathbf{x}_1, \ldots, \mathbf{x}_N$ with $\mathbf{x}_i \in \mathbb{R}^d$ and corresponding label information $y_1, \ldots, y_N$ with $y_i \in \{-1, 1\}$, the task in supervised classification is to train a model f for the prediction of the label of an unknown pattern $\mathbf{x}' \in \mathbb{R}^d$. SVMs are successful models for such supervised learning tasks. They are based on maximizing the margin of a hyperplane that separates patterns of different classes. The dual SVM optimization problem is to maximize

$$L_d = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{N}\alpha_i \tag{1}$$

with respect to α_i subject to constraints $\sum_{i=1}^{N}\alpha_i y_i = 0$ and $\alpha_i \geq 0 \ \forall i$. Once the optimization problem is solved for α_i^*, in many scenarios the majority of patterns vanish with $\alpha_i = 0$ and only few have $\alpha_i > 0$. Patterns $\mathbf{x}_i$, for which $\alpha_i > 0$ holds, are called *support vectors*. The separating hyperplane $\mathbf{H}$ is defined with these support vectors:

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i. \tag{2}$$

The support vectors satisfy $y_i(\mathbf{w}^T\mathbf{x}_i + w_0) = 1$, while lying on the corner of the margin. With any support vector $\mathbf{x}_i$, we can compute $w_0 = y_i - \mathbf{w}^T\mathbf{x}_i$ and the resulting discriminant $f(\mathbf{x}') = \text{sign}(\mathbf{w}^T\mathbf{x}' + w_0)$, which is called SVM. An SVM that is trained with the support vectors computes the same discriminant function as the SVM trained on the original training set, a principle that is used in cascade SVMs. Extensions of SVMs have been proposed that allow learning with large data sets such as core vector machines [5].

2.2. Cascade SVMs. The classic cascade SVM (C-SVM) [2] employs horizontal cascading; that is, it divides the training set into smaller subsets, which can be computed more efficiently. The idea to divide the problem into smaller subproblems came up early, for example, by Jacobs et al. [6]. Recently, Hsieh et al. [7] proposed a divide-and-conquer solver showing that the support vectors identified by the subproblem solution are likely to be support vectors of the entire kernel SVM problem based on an adaptive clustering approach.

In the first step of C-SVM learning, an intermediate training set size n and a target training set size n^* (corresponding to the final number of support vectors) have to be defined. We employ the following cascade variant; see Figure 1. The training set $T_{[1,N]} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ of patterns $\mathbf{x}_i$ with corresponding labels y_i is divided into subsets of size n; that is,

$$T_{[j,k]} = \left\{\left(\mathbf{x}_j, y_j\right), \ldots, \left(\mathbf{x}_k, y_k\right)\right\}, \tag{3}$$

with $\mathbf{x}_j, \ldots, \mathbf{x}_k \in \mathbb{R}^d$ and $y_j, \ldots, y_k \in \{-1, 1\}$. On the first training set $T_{[1,n]}$, the SVM parameters (kernel type, bandwidth parameter, and regularization parameters) are chosen with grid search and cross-validation that are subject to the complete SVM parameter setup. This optimal parameter set κ^* is used for all SVMs.

Each SVM returns the support vectors as learning result. In each iteration, the training set is reduced to the corresponding set of support vectors. This process is stopped, when the final number of support vectors is smaller or equal to n^*. The resulting support vector set is the basis of the final SVM that can be employed as final estimator.

For C-SVMs, the reduction of runtime on the first level becomes $N^3 > (N/n) \cdot n^3 = N \cdot n^2$. A similar argument holds for all subsequent levels. The speedup can be increased by parallelizing the training process on multicore machines. However, small cascade training set sizes n result in larger support vector sets for each level and consequently more cascade levels.

Figure 2 illustrates the learning results of a C-SVM with radial basis function (RBF) kernel that divides the training set of patterns from the XOR problem into two parts. Figure 2(a) shows the learning result of an SVM with RBF kernel. Figure 2(b) shows the learning result of an SVM trained with the support vectors computed by an SVM that has been trained on the first half of the XOR data set. The corresponding SVM trained with the support vectors of the second half of the data set is shown in Figure 2(c). Figure 2(d) shows the decision boundary of the C-SVM trained with the union of both support vector sets. The figures show that the original SVM and the C-SVM learn the same decision boundary.

2.3. Extreme Cascading. Often, not all features are important to efficiently solve classification problems; some may be uncorrelated with the label or redundant. The reduction of the number of features to a relevant subset in supervised learning is a common approach in machine learning [8, 9]. Plastria et al. [10] have shown that a proper choice of methods and number of dimensions can yield a significant increase in classifier performance.

The extreme cascade model we propose in this work combines cascading with dimensionality reduction. Algorithm 1 shows the pseudocode of the ECM approach. Each training subset is subject to a DR preprocessing resulting in reduced q-dimensional subsets $\widehat{T}_{[j,k]} = \{(\widehat{\mathbf{x}}_j, y_j), \ldots, (\widehat{\mathbf{x}}_k, y_k)\}$ with $\widehat{\mathbf{x}}_{j,\ldots,k} \in \mathbb{R}^q$. The reduced training sets $\widehat{T}_{[\cdot,\cdot]}$ are each subject to

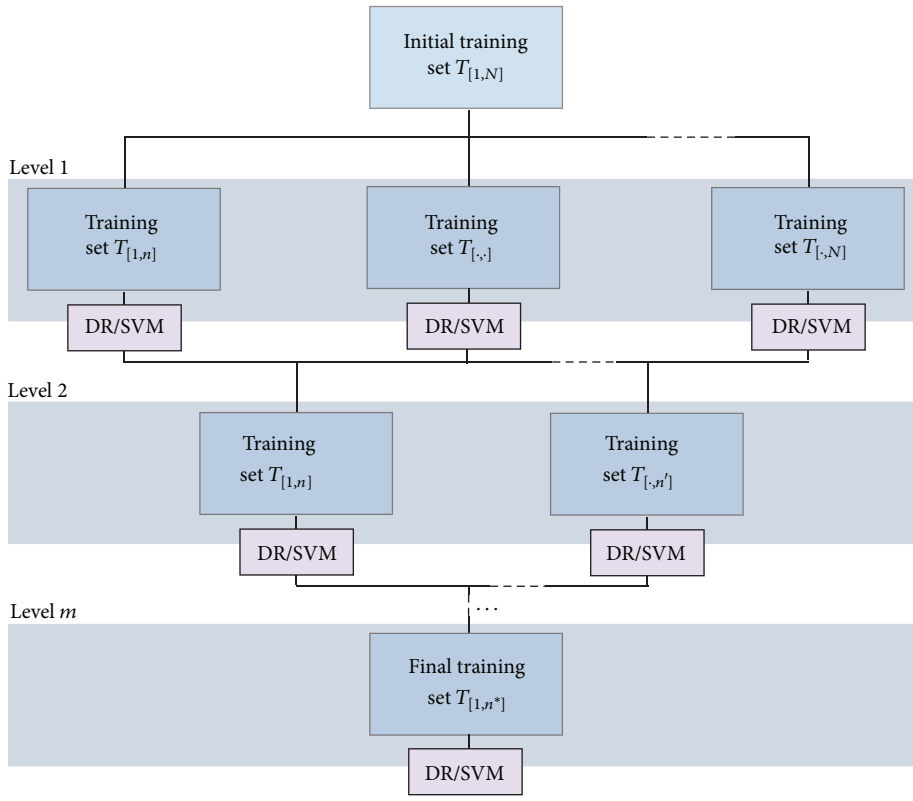


Figure 1: Illustration of C-SVM and ECM learning scheme. In each level, the training set is divided into N/n subsets. This procedure is repeated consecutively until the target size n^* of training set for support vectors is reached. In case of ECM, the SVM learning process in the low-dimensional space returns low-dimensional support vectors. In each next level, the high-dimensional pendants are employed.

SVM training. After the SVM training in the q-dimensional space, the patterns $\mathbf{x}_i \in \mathbb{R}^d$ corresponding to the support vectors $\hat{\mathbf{x}}_i \in \mathbb{R}^q$ of the intermediate SVMs, that is, with $y_i(\widehat{\mathbf{w}}^T\hat{\mathbf{x}}_i + \widehat{w}_0) = 1$, are collected in the support vector set T'. As the SVM training time depends on the pattern dimensionality, the reduction to values $q \ll d$ usually accelerates the computation time. The assumption of ECM is that the dimensionality reduction process maintains most properties of the high-dimensional data space and that the set of support vectors of the low-dimensional space is similar to the support vector set of the high-dimensional data. This assumption will be analyzed in the experimental part.

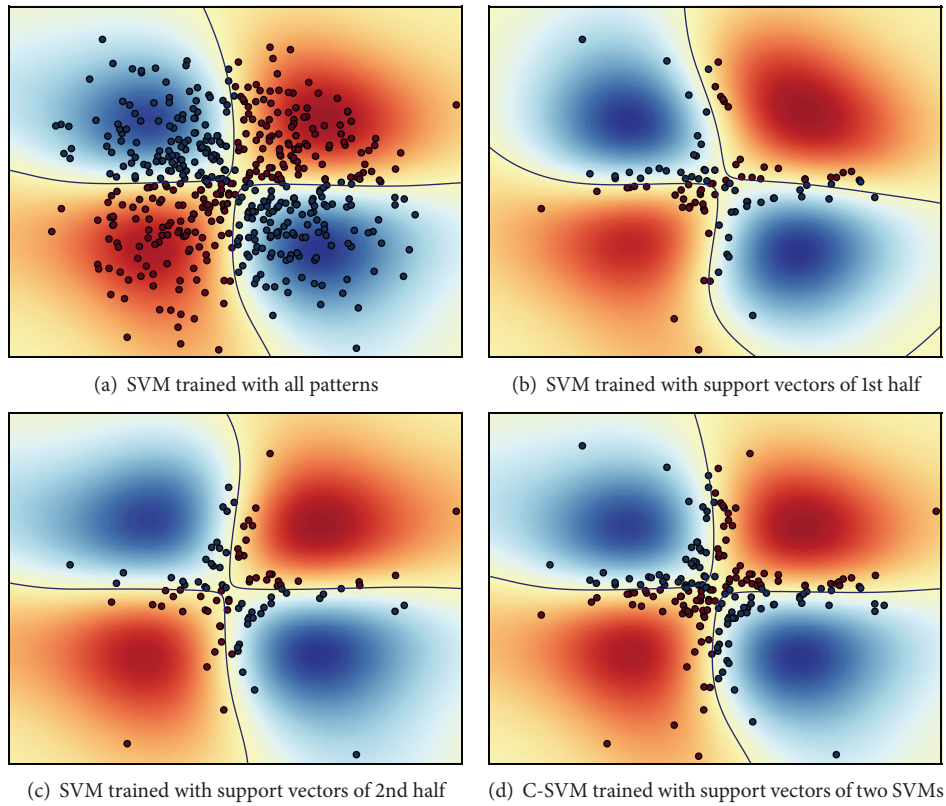
The final SVM* is trained on the last training set T consisting of the patterns $\hat{\mathbf{x}}_i \in \mathbb{R}^q$ that correspond to the support vectors of the last cascade level. The ECM employs many parameters that can be tuned to define the complete ECM model, from DR choice and target dimensionality q to DR method parameters and SVM parameters like kernel type, bandwidth parameters, and regularization parameter C. Some of the parameters are analyzed in the following experimental part of this work.

3. Experimental Analysis

In the following, we analyze the ECM variants experimentally concentrating on principal component analysis (PCA) [11], isometric mapping (ISOMAP) [12], and locally linear embedding (LLE) [13] for preprocessing.

3.1. Support Vector Analysis. We start the experimental part with an analysis of the assumption that the support vectors learned by the SVM in the low-dimensional space are the same as the support vectors an SVM learns in the high-dimensional space. Figure 3 shows the ratio of common support vectors of both SVMs and the number of support vectors of the SVM in the high-dimensional space with respect to an increasing dimensionality q of the low-dimensional space. One curve shows the average, maximum, and minimum ratios of 20 runs with new instances of the MakeClass data set (cf. Appendix for a detailed description) with increasing q. The blue curve shows the results for less structured instances of MakeClass with $\Delta = 0.15$, while the red curve employs more informative features with $\Delta = 0.2$. As expected, we can observe that the ratio of common support vectors increases with the dimensionality of the low-dimensional space until a q is reached, as of which the ratio remains 1.0. This state is reached later for $\Delta = 0.2$, as more support vectors are necessary for data sets that employ more informative features and less redundancy.

3.2. Parameter Analysis. The ECM approach depends on a proper choice of parameters. We analyze parameter settings



(a) SVM trained with all patterns

(b) SVM trained with support vectors of 1st half

(c) SVM trained with support vectors of 2nd half

(d) C-SVM trained with support vectors of two SVMs

Figure 2: Comparison of SVM learning results of (a) a classic SVM trained on a set of patterns (XOR data set), (b) an SVM trained with the support vectors of an SVM that has been trained on the 1st half of the XOR data set, (c) an SVM trained on the support vectors, 2nd half of XOR data set, and (d) the C-SVM trained on the support vectors of both SVMs.

Table 1: Comparison of ISOMAP-ECM and LLE-ECM with various choices for q and k in terms of MSE on MakeClass data set with N = 100,000, d = 40, and Δ = 0.3.

DR	ISOMAP						LLE					
k	10		20		30		10		20		30	
q	δ	t	δ	t	δ	t	δ	t	δ	t	δ	t
10	0.03	46.88	**0.02**	67.17	0.03	138.60	0.03	156.78	**0.02**	*171.14*	0.03	171.80
20	0.03	48.01	**0.02**	*66.41*	**0.02**	96.00	0.03	170.64	0.03	184.79	0.03	186.10
30	0.04	94.43	0.03	65.01	0.03	91.91	0.04	177.35	0.03	184.00	0.03	183.21

for the PCA-ECM in Figure 4(a) on the MakeClass [14] data set with N = 30,000, d = 100, and Δ = 0.2. To set the SVM parameters C and σ for the RBF kernel, we employ a 5-fold cross-validation in all the following experiments. The plot shows accuracy and training time with various settings for q, n, and n^*. We can observe that the accuracy of all ECM variants lies between 0.9 and 0.99. The highest accuracies have been achieved with setting q = 40, while the fastest run with a high accuracy has been achieved with settings q = 40, n = 3,000, and n^* = 5,000. This cascade is comparatively sensitive concerning the choice of q. A similar comparison is shown in Figure 4(b), where LLE-ECM and ISOMAP-ECM variants with various settings are compared on the MakeClass data set. The results show that ISOMAP is a faster preprocessing method and often achieves high accuracies, but it is at the same time less robust than LLE, as bad parameter choices can result in comparatively low classification accuracies.

For a closer look at the LLE and ISOMAP variants, Table 1 shows runtime and mean squared error (MSE) results on the MakeClass data set with N = 100,000, d = 40, and Δ = 0.3 and ECM settings n = 1,000 and n^* = 5,000. Various settings for neighborhood size k of LLE and ISOMAP and for target dimensionality q are employed. The results show that the approaches are faster for smaller k, an effect that is more significant for ISOMAP than for LLE. Surprisingly, we can observe a tendency for better accuracies, if smaller values for q are employed. This may be due to the fact that the manifold learning process is forced to concentrate on the most important features of the data set, while higher dimensions introduce noise to the classification problem.

Require: training set T, n, n^*
(1) $\kappa^* \leftarrow$ cv/grid search on $T_{[1,n]}$
(2) **repeat**
(3) **for** $\alpha = 1$ **to** $N/n - 1$ **do**
(4) select training set $T_{[\alpha \cdot n,(\alpha+1)\cdot n]}$
(5) reduce dimensionality $\widehat{T}_{[\alpha \cdot n,(\alpha+1)\cdot n]}$
(6) get support vectors $\{\widehat{\mathbf{x}}_i \in \mathbb{R}^q\}$
(7) train SVM on $\widehat{T}_{[\alpha \cdot n,(\alpha+1)\cdot n]}$
(8) $T' \rightarrow T' \cup \{\mathbf{x}_i \in \mathbb{R}^d\}$ (pendants of $\{\widehat{\mathbf{x}}_i \in \mathbb{R}^q\}$)
(9) **end for**
(10) $T = T', T' = \{\}$
(11) **until** $|T| \leq n^*$
(12) train SVM* on T
(13) **return** T, SVM*

Algorithm 1: Pseudocode of ECM algorithm. In each cascade level, the d-dimensional patterns $\mathbf{x}_i$, which are the pendants of the q-dimensional support vectors $\widehat{\mathbf{x}}_i \in \mathbb{R}^q$ of the previous cascade level, are employed as training patterns.

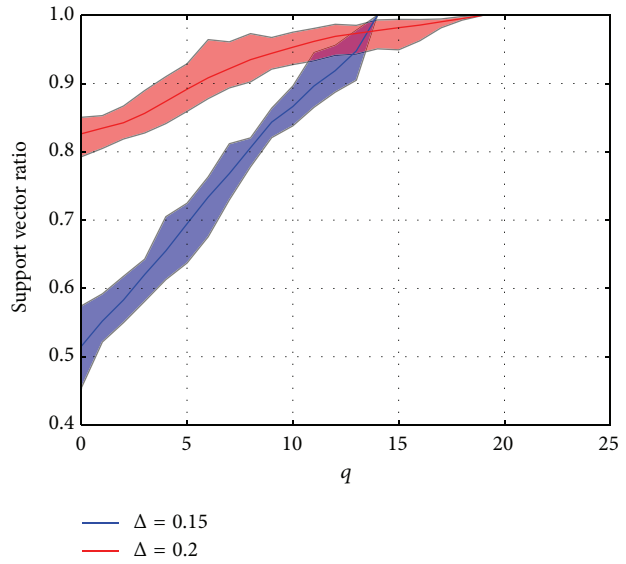
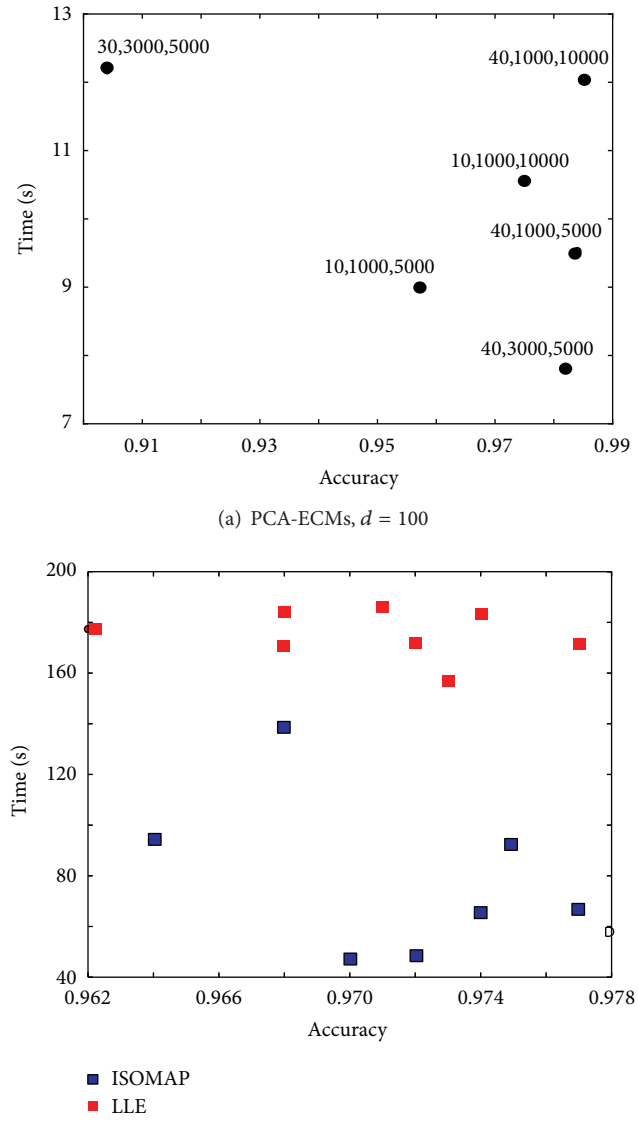


Figure 3: Analysis of support vector consistency with respect to target dimensionality q on two instances of MakeClass ($\Delta = 0.15$, $\Delta = 0.2$).



(a) PCA-ECMs, $d = 100$

(b) LLE- and ISOMAP-ECMs, $d = 100$

Figure 4: Illustration of ECM classifier accuracies and training times on MakeClass data set: (a) various PCA-ECMs with settings (q, n, n^*) on data set with $N = 30{,}000$, $d = 40$, and $\Delta = 0.2$ and (b) a comparison between LLE- and ISOMAP-ECMs.

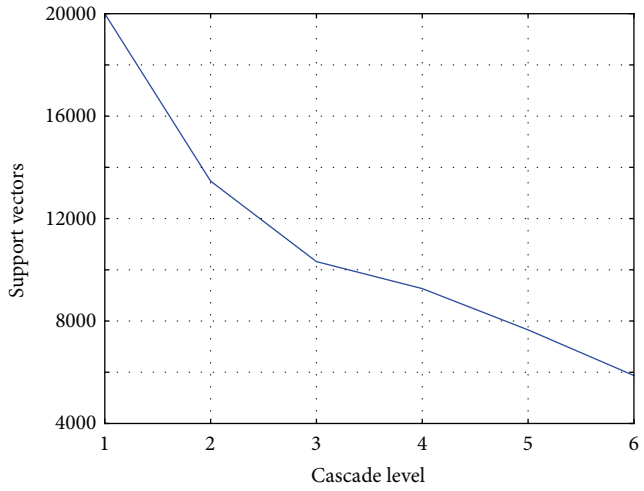
The best result (highest accuracy with best runtime) has been achieved by ISOMAP with settings $q = 20$ and $k = 20$.

A typical development of the number of support vectors during a typical run is shown in Figure 5(a). The number of support vectors is decreasing approximately linearly in the course of successive cascade levels. Figure 5(b) shows a runtime comparison of SVM and ECM on an increasing training set size of MakeClass with $N = 30{,}000$, $n = 1{,}000$ and ECM settings $n^* = 5{,}000$ with $d = 20$ and $\Delta = 0.2$. The training time of the SVM is exponentially increasing, while the ECM training time is increasing moderately. For example, the ECM takes $t = 13$ s and the SVM takes $t = 70$ s employing a training set of size $N = 50{,}000$.

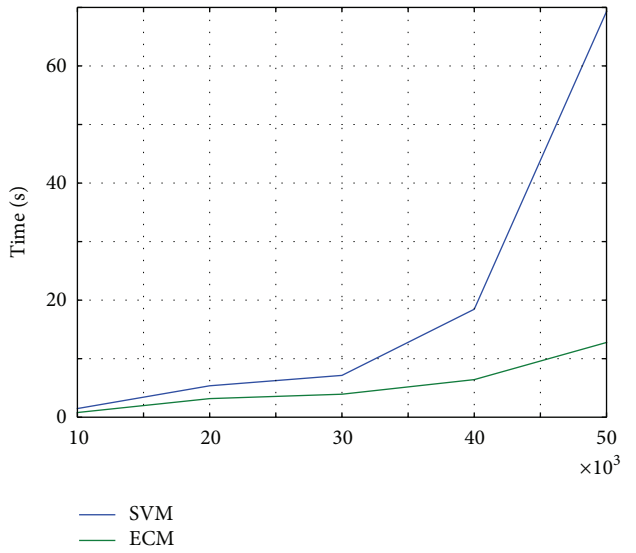
3.3. Benchmark Data Sets. As the PCA-ECM is the fastest ECM variant with strong accuracies, we concentrate on a comparison of PCA-ECM, C-SVM, and a standard SVM on a larger benchmark data set; see Appendix, in the following. The benchmark data set contains the four classification problems MakeClass, Hastie, Faces, and Blobs. The problems Friedman 1, Friedman 3, and Wind are regression problems. Table 2 shows the test error δ (normalized MSE) after training on a training set achieved on an independent test set and the runtime (the runtime depends on the machine (2.7 GHz Intel Core i5), the operating system (Apple OS X), and the programming language and packages (Python and scikit-learn)) t of training and test phase of the three classifiers.

Table 2: Comparison between PCA-ECM, C-SVM, and SVM in terms of MSE on benchmark problems.

Problem	Parameters			PCA-ECM		C-SVM		SVM	
	q	n	n^*	δ	t	δ	t	δ	t
MakeClass	5	1000	5000	0.041	12.930	0.037	25.960	0.022	72.048
Hastie	2	500	1000	0.025	18.095	0.036	8.154	0.002	46.762
Faces	5	200	600	0.145	2.151	0.165	3.899	0.140	3.930
Blobs	5	1000	5000	0.194	8.287	0.468	16.575	0.190	50.469
Friedman 1	5	1000	5000	0.077	45.570	0.078	40.057	0.051	357.690
Friedman 3	3	1000	5000	5.134	32.940	5.110	58.915	3.403	910.852
Wind	2	500	1000	9.533	29.118	9.599	37.035	8.787	87.607



(a) PCA-ECMs, $d = 40$



(b) SVM and ECM training time on increasing training set size

Figure 5: Illustration of ECM training characteristics: (a) development of the number of support vectors during a typical run; the number of support vectors is decreasing approximately linearly and (b) runtime comparison of SVM and ECM on an increasing training set size with $n = 1{,}000$ and $n^* = 5{,}000$ on MakeClass with $d = 20$ and $\Delta = 0.2$.

The cascade classifiers PCA-ECM and C-SVM employ the settings for n and n^*; PCA-ECM uses the specified latent space dimensionality q. These parameters have been found in manual and automatic tuning processes (with grid search), but we tried to use similar settings to draw parameter-independent conclusions.

The MakeClass benchmark problem is an artificial data set with setting $\Delta = 0.22$ for balance between informative and redundant features and training set size $N = 50{,}000$ with $d = 100$. PCA-ECM turns out to be the fastest variant, while the classic SVM is the slowest but achieves lowest error. PCA-ECM achieves a larger error. The accuracy dropout of the PCA-ECM may be acceptable (2% in accuracy, from 0.022 to 0.041), when considering the fact that the PCA only requires 18% of the SVM runtime. The Hastie data set is an artificial data set from Hastie et al. [8], with $N = 100{,}000$ and $d = 10$, which is not high-dimensional, but large. Here, the SVM is still the fastest classifier, but the situation changes when we compare PCA-ECM and C-SVM. Now, the PCA-ECM is faster, but the C-SVM achieves lower test error.

On Faces, the PCA-ECM achieves a higher accuracy than the C-SVM but is slightly worse than the classic SVM. Although the SVM is already fast, the PCA-ECM is slightly faster. On the Blobs data set, DR-based preprocessing seems to be important as the C-SVM completely fails, while the PCA-ECM achieves very good results, almost as good as the classic SVM. The latter is about six times slower than the PCA-ECM. On the regression data sets, a similar picture can be drawn. The PCA-ECM achieves a lower error than the C-SVM on Friedman 1. Although the SVM is better again, it requires almost six minutes to compute the solution. On Friedman 3, the situation is similar. SVM achieves the lowest error but requires much more time than the cascade variants, which are only slightly worse in accuracy. On the Wind data set, a similar observation can be made like that on Friedman 1, Hastie, Faces, and Blobs. The DR process introduces an advantage in accuracy and also in runtime. Taking into account all seven benchmark data sets, we can observe that the SVM always achieves lowest test error but requires the longest training and test time. The cascade variants are always faster than the classic SVM with the expected tradeoff concerning the accuracy.

4. Conclusions

ECMs allow the application of SVMs to large data sets by cascading training sets and at the same time reducing the dimensionality of patterns. We experimentally analyzed fast variants that are based on preprocessing with DR and C-SVMs, in particular PCA-based ECMs concentrating on parameters like intermediate and final cascade training set size. Both parameters have a significant influence on the final classification result. Further comparisons of DR methods for preprocessing have shown that ISOMAP outperforms LLE in terms of runtime. Most cascade variants lead to a fast training process while maintaining the same classifier accuracy or only paying with a slight decrease of the accuracy. As the computation of intermediate SVMs can be parallelized on each level, the distribution to multiple cores allows a further significant speedup. This will be subject to future investigations. Further, an extensive comparison to core SVMs, which achieve the speedup based on approximations using a minimum enclosing ball [5], will allow a rigorous comparison of SVMs for large data sets.

Appendix

Benchmark Problems

In the following, the employed benchmark data sets are shortly described. We employ a data set size of N and employ the last N_t patterns for the test set.

(i) MakeClass is a classification data set ($N = 50000$, $N_t = 1000$, and $d = 100$) generated with the SCIKIT-LEARN [14] method make_classification with $d = 100$ dimensions and two centers. The structure Δ determines the ratio informative features; the remaining ones are redundant.

(ii) The data set Hastie ($N = 20000$, $N_t = 1000$, and $d = 100$) generates data for binary classification and has been employed in Hastie et al. [8]. It can be generated with the SCIKIT-LEARN [14] method make_hastie_10_2.

(iii) The Faces data set is called *Labeled Faces in the Wild* [15] ($N = 1088$, $N_t = 200$, and $d = 1850$) and has been introduced for studying the face recognition problem. The data set source is http://vis-www.cs.umass.edu/lfw/. It contains JPEG images of famous people collected from the internet. The faces are labeled with the name of the person pictured.

(iv) The Gaussian Blobs data set ($N = 20000$, $N_t = 1000$, and $d = 100$) is generated with the SCIKIT-LEARN [14] method make_blobs and the following settings. Two centers, that is, two classes, are generated, each with a standard deviation of $\sigma = 10.0$ and variable d.

(v) Friedman 1 ($N = 20000$, $N_t = 1000$, and $d = 100$) is a regression data set generated with the SCIKIT-LEARN [14] method make_friedman1. The regression problem has been introduced in [16], where Friedman introduces multivariate adaptive regression splines.

(vi) Friedman 3 ($N = 50000$, $N_t = 1000$, and $d = 4$) is also a regression data set of SCIKIT-LEARN [14] and can be generated with make_friedman3.

(vii) The Wind data set ($N = 50000$, $N_t = 5000$, and $d = 11$) is based on spatiotemporal time series data from the National Renewable Energy Laboratory (NREL) Western Wind data set. The whole data set comprises time series of 32,043 wind turbines, each holding ten 3 MW turbines over a timespan of three years in a 10-minute resolution. The dimensionality is $d = 22$.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The author thanks NREL for publication of the Wind data set that is part of his experimental comparison.

References

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[2] H. P. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, and V. Vapnik, "Parallel support vector machines: the cascade svm," in *Advances in Neural Information Processing Systems*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17, pp. 521–528, MIT Press, Cambridge, Mass, USA, 2005.

[3] R. Ye, Z. Le, and P. N. Suganthan, "K-nearest neighbor based bagging SVM pruning," in *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL '13)*, pp. 25–30, Singapore, April 2013.

[4] N. Gianniotis and C. Riggelsen, "Visualisation of high-dimensional data using an ensemble of neural networks," in *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL '13)*, pp. 17–24, Singapore, April 2013.

[5] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: fast SVM training on very large data sets," *Journal of Machine Learning Research*, vol. 6, pp. 363–392, 2005.

[6] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[7] C.-J. Hsieh, S. Si, and I. S. Dhillon, "A divide-and-conquer solver for kernel support vector machines," in *Proceedings of the International Conference on Machine Learning (ICML '14)*, pp. 566–574, 2014.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer, New York, NY, USA, Second edition, 2009.

[9] J. Yan, B. Zhang, N. Liu et al., "Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 320–332, 2006.

[10] F. Plastria, S. D. Bruyne, and E. Carrizosa, "Dimensionality reduction for classification," in *Advanced Data Mining and Applications*, vol. 5139 of *Lecture Notes in Computer Science*, pp. 411–418, Springer, Berlin, Germany, 2008.

[11] I. Jolliffe, *Principal Component Analysis*, Springer Series in Statistics, Springer, New York, NY, USA, 1986.

[12] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[13] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: a database for studying face recognition in unconstrained environments," Tech. Rep. 07-49, University of Massachusetts, Amherst, Mass, USA, 2007.

[16] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.