SVMs and Data Dependent Distance Metric

N. Zaidi, D. Squire

Clayton School of Information Technology, Monash University, Clayton, VIC 3800, Australia
Email: {nayyar.zaidi,david.squire}@monash.edu

Abstract

Support Vector Machine (SVM) is an efficient classification tool. Based on the principle of structured risk minimization, SVM is designed to generalize well. But it has been shown that SVM is not immune to the curse of dimensionality. Also SVM performance is not only critical to the choice of kernel but also to the kernel parameters which are generally tuned through computationally expensive cross-validation procedures. Typical kernels do not have any information about the subspace to ignore irrelevant features or making relevant features explicit. Recently, a lot of progress has been made for learning a data dependent distance metric for improving the efficiency of k-Nearest Neighbor (KNN) classifier. Metric learning approaches have not been investigated in the context of SVM. In this paper, we study the impact of learning a data dependent distance metric on classification performance of an SVM classifier. Our novel approach in this paper is a formulation relying on a simple Mean Square Error (MSE) gradient based metric learning method to tune kernel's parameters. Experiments are conducted on major UCIML, faces and digit databases. We have found that tuning kernel parameters through a metric learning approach can improve the classification performance of an SVM classifier.

Keywords: local methods, metric learning, support vector machine, Gaussian kernel tuning, object recognition.

1 Introduction

Support Vector Machine (SVM) classifiers have been shown to give state of the art performance on a wide range of classification data sets. An SVM finds an optimal hyperplane to separate data into two classes. Given training data set $\{(\vec{x}_1, y_1), ...(\vec{x}_n, y_n)\}$, the decision function is found by solving the following convex optimization problem (Lagrangian Dual):

$$\begin{aligned} \max_{\alpha} f(\alpha) \quad &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\vec{x}_i, \vec{x}_j) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \mathcal{C} \quad \text{and} \quad \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \\ \text{where} \quad & k(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}\right) \end{aligned} \quad (1)$$

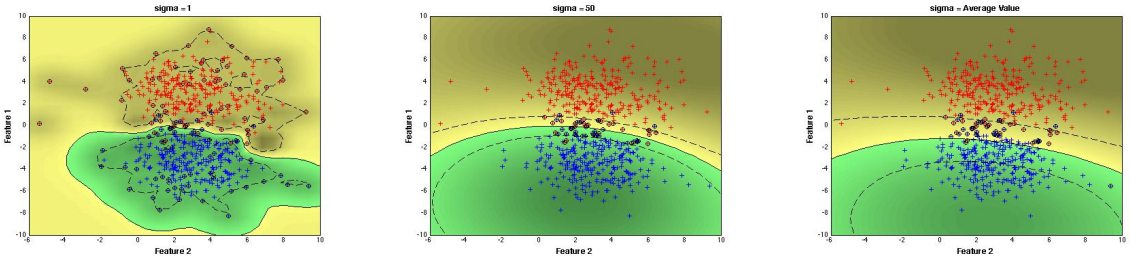
where α are the Lagrange coeficients, $\mathcal{C}$ controls the misclassification penalty and $k(.,.)$ is the kernel function. It is a widely acknowledged fact that SVM 's performance is critically dependent on the kernel's choice and its parameters [1]. Since different kernels (also the same kernel with different parameters) embed data in a different high dimensional space, a linear decision boundary which SVM finds in that space varies by the choice of kernel and its parameters, hence affecting its performance.

978-1-4244-9631-0/10/$26.00 ©2010 IEEE

One of the widely used kernel is the Gaussian kernel (equation 1) who's parameters are typically tuned through expensive cross-validation procedure. It should be noted that as no prior knowledge is available about the meaning of the attributes, for sheer simplicity, the kernel is assumed to be isotropic. That is all attributes are given the same weight ($\sigma = \sigma_1 = ... = \sigma_d$ for all d features). But in most real-world databases, attributes are of very different nature and there is a need to scale features so that each feature is given the appropriate weight. Adapting the shape of the kernel is useful as it not only results in building the knowledge about the data into the kernel but also results in feature selection [2]. Feature selection is extremely desirable for removing the curse of dimensionality [1] (COD).

The need to choose σ parameter is also critical as the kernel choice is actually a regularization choice and scaling parameters (σ) of the kernel (often known as the smoothing parameters) controls the bias and variance of the classifier by controlling the extent of smoothing [1, sec 7.8]. Having a large value of σ results in an loosely fit function, whereas a small value may result in an over-fit (figure 1

[1] COD refers to the fact that any feasible number of training data only sparsely populate the input space. Due to this, we will never have enough training data in high dimensions to make learning algorithms robust.



(a) (b) (c)

Figure 1: Demonstration of the impact of varying the σ parameter of a Gaussian kernel in support vector machine framework on the synthetic data. It can be seen that Feature 2 (x-axis) is redundant and data can be well separated with Feature 1 (y-axis) only. 1(a): $\sigma = 1$, the classifier is over-fitted. The classifier has memorized the training data. 1(b): $\sigma = 50$, the classifier is loosely fitted to the training data. 1(c): $\sigma =$ averaged distance of N nearest neighbors (equation 2). Setting σ to be the average value of the distances between the training data seems like a sensible strategy if no information is present about the features.

and 2). Typically in computer vision research [3, 4] to avoid expensive cross-valiation, σ is set to be the average value of the squared distance between all data points, as shown in the following equation:

$$\sigma^2 = \frac{1}{N} \sum_{i,j=1}^{N} d^2(\vec{x}_i, \vec{x}_j) \tag{2}$$

This strategy not only ensures that the kernel's numeric range is within a bounded interval but also results in achieving an optimal trade-off between bias and variance of the classifier (figure 1(c)). It has been shown that SVM classifiers are not immune to the curse of dimensionality [5, sec 12.3.4]. So assuming an isotropic kernel may not be optimal. Also using σ as the averaged value of distance between the entire training data may not be optimal due to the presence of irrelevant features. Consider the impact of an an-isotropic kernel on contrived data in figure 2. Feature 2 is redundant and $\sigma = [0.25\ 10]'$ gives a fine optimal decision boundary.

Recently a lot of work has been done in the area of metric learning to improve the performance of

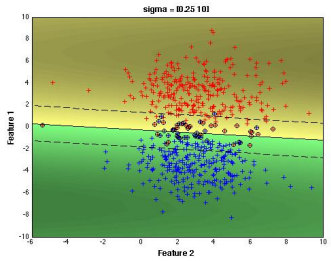


Figure 2: An an-isotropic Guassian kernel with $\sigma = [0.25\ 10]'$ is used. The classifier has ignored Feature 2 from consideration and resulting boundaries are more linear and better fits the training data.

k-nearest neighbor (k-NN) classifier [6, 7, 8, 9, 10]. Metric learning algorithms are aimed at finding a metric that results in small intra-class and large inter-class distance. Metric is parametrized by a norm and a positive semi-definite matrix. Typically an inverse square root of the distance matrix is estimated. That is we learn a matrix parmeterizing the linear transformation of the input space such that in the transformed space k-NN performs well. The goal of this work is to study the effect of metric learning on SVM. To the best of our knowledge, metric learning in the context of SVM has not been investigated. In the following, we will establish how learning a data dependent distance metric can impact SVM classification performance. As can be seen, the kernel in equation 1 can be written as:

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(\frac{-d^2(\vec{x}_i, \vec{x}_j)}{2\sigma^2}\right) \tag{3}$$

In the lights of metric learning, $d^2(\vec{x}_i, \vec{x}_j)$ in equation 3 can be replaced by a more general metric L: that is $d^2_L(\vec{x}_i, \vec{x}_j)$. If $L = A^T A$, then $d^2_L(\vec{x}_i, \vec{x}_j) = (A\vec{x}_i - A\vec{x}_j)^T(A\vec{x}_i - A\vec{x}_j)$. It is more helpful to optimize A rather than L, because optimization for L requires to fulfill semi-positive constraint which is expensive to maintain. The kernel in equation 3 can be written as:

$$k(\vec{x}_i, \vec{x}_j) = \exp\left(-\|A\vec{x}_i - A\vec{x}_j\|_2^2\right) \tag{4}$$

This suggests that metric learning is in fact kernel learning and as discussed, kernel controls SVM performance. Therefore, learning a data dependent distance metric can improve support vector machine's classification performance. Any metric learning method that gives a linear transformation of data by learning the matrix A can be used. We have used a simple metric learning approach based on the gradient minimization of Mean Square Error

(MSE) in k-NN framework. Other metric learning methods for example neighborhood component analysis [6], large margin nearest neighbor [8] algorithms etc can be used.

2 Approach

In this section, we will describe our proposed algorithms to train SVM with data dependent distance metric. We have used MEGM metric learning algorithm from [11] for learning the kernel parameters. We will briefly explain MEGM (Mean Square Error's Gradient Minimization) metric learning algorithm in the following section and then describe our two proposed algorithms.

2.1 MEGM

In a typical regression setting, an unknown function $f : R^D \rightarrow R$ is predicted from the training data $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_N, y_N)\}$, where $\vec{x}_i$ is a data point and y is the corresponding target value. The predicted function $\hat{f}$ is chosen to be the one that minimizes some loss function such as 'mean square error' (MSE) etc. For classification task having C classes, the MSE for a data set containing N number of points is given as:

$$\text{MSE}(\hat{y}) = \sum_{t=1}^{C} \sum_{i=1}^{N} (y_{ti} - \hat{y}_{ti}) \tag{5}$$

where $\hat{y}_i$ denotes the predicted probability of point $\vec{x}_i$ and y_i denotes its true class label (either 0 or 1). For brevity we have denoted $\hat{y}(\vec{x}_{ti})$ with $\hat{y}_{ti}$ and $y(\vec{x}_{ti})$ with y_{ti}. In the following discussion, we will assume that there are only two classes to make our derivations simple. For any query point $\vec{x}_i$, nearest neighbor methods work by predicting the value $\hat{y}_i$ by considering the labels of its k nearest neighbors. In order to have a smooth boundary, each neighbor votes for the query label based on its distance from the query point [5]. Equation 6 shows the Nadaraya-Watson kernel for regression:

$$\hat{y}(\vec{x}_i) = \frac{\sum_j y_j V_j}{\sum_j V_j} \tag{6}$$

The vote V_j casted by each label around the query point $\vec{x}_i$ is usually chosen to be a function that decays exponentially as the distance from the query point increases, for example, Gaussian kernel (equation 3). Determining votes using equation 6 assumes a well defined distance measure. This assumption is not always true due to the reasons like curse of dimensionality, presence of irrelevant features etc and can lead to bad results. As described in section 1, $d^2(\vec{x}_i, \vec{x}_j)$ in the formulation of Gaussian kernels can be replaced by a more general metric that is: $d_L^2(\vec{x}_i, \vec{x}_j)$ where $L = A^T A$. Since MSE is a function of $\hat{y}$ and $\hat{y}$ depends on $||\vec{x}_i - \vec{x}_j||_L^2$, MSE can be minimized by selecting an optimal value of L. In other words, a change in L induces a change in the distance, which can alter the votes. This alteration in the votes (V_j) triggers a change in $\hat{y}$ affecting the MSE. Obviously trying all possible values of L is not feasible. Some sort of search mechanism is required to find an optimal value of L. Votes V_j can be replaced by W_j as:

$$W_j = \exp\left(\frac{-\|A\vec{x} - A\vec{x_j}\|_2^2}{2\sigma^2}\right) \tag{7}$$

MEGM metric learning method is based on gradient descent algorithm to minimize MSE (let us denote MSE by E). The gradient of E with respect to the matrix A is shown in equation 8. This gradient is minimized to get an optimal metric that is parameterized by the matrix A. Convergence to the global minimum is not guaranteed. The risk of local minima can be reduced by running the algorithm several times with different training samples and choosing the output with the minimum error (E).

2.2 Globally Adaptive SVM (GASVM)

The outline of our proposed algorithms Globally Adaptive SVM classifiers GASVM1 and GASVM2 are given in algorithm 1 and 2.

The proposed GASVM1 algorithm tunes the kernel by learning a data dependent distance metric using MEGM algorithm before training an SVM classifier. C SVM classifiers needs to be trained for C classes (with one versus all strategy). Once the matrix L is learned, the kernel k_L is used by all C SVM classifiers. GASVM2 is a slight variant of GASVM1. That is, rather than learning one kernel for all the categories, GASVM2 learns a different kernel using MEGM algorithm for each category. The learned kernel k_{Lc} is saved with the classifier for category c. Whenever the classifier c is used to classify the test point ($\vec{x}_0$), it uses the learned kernel (metric) k_{Lc} to measure the similarity. As will be shown in section 3, in terms of the classification performance, both GASVM1 and GASVM2 perform equally well on different data sets. Since GASVM1 uses only one kernel for all classifiers, it is more computational efficient than GASVM2.

Though other kernels can be incorporated in GASVM formulation, the scope of the work is limited to the Gaussian kernels. Since Gaussian kernels have been shown to perform extremely well on a huge variety of data sets and are widely used, this should not be considered as a drawback. However, those cases where similarity is more efficiently measured by some other kernel for example linear or polynomial, the use of GASVM algorithms may not be beneficial.

$$\frac{\partial E}{\partial A} = 2A(y_i - \hat{y}_i)\frac{1}{\sum_j W_j}\sum_j (y_j - \hat{y}_j)W_j(\vec{x} - \vec{x}_j)(\vec{x} - \vec{x}_j)^T \quad (8)$$

Algorithm 1 GASVM1: Train an SVM classifier using a data dependent distance metric.

Require:
- $\vec{x}_0$: Testing data.
- $\{\vec{x}_n, y_n\}_{n=1}^N$: Training data.

- Get a data dependent distance metric L using MEGM metric learning algorithm such that $L = A^T A$.

for $c = 1, 2, \ldots, C$ **do**
- Train an SVM classifier for category c using the kernel:

$$k_L(\vec{x}_i, \vec{x}_j) = \exp\left(-\|A\vec{x}_i - A\vec{x}_j\|_2^2\right)$$

end for
- Use C SVM classifiers in one-versus-all way to classify $\vec{x}_0$ using the kernel k_L.

Algorithm 2 GASVM2: Train an SVM classifier using a data dependent distance metric.

Require:
- Testing data: $\vec{x}_0$.
- Training data: $\{\vec{x}_n, y_n\}_{n=1}^N$.

for $c = 1, 2, \ldots, C$ **do**
- Get a data dependent distance metric L using MEGM metric learning algorithm for category c such that $L = A^T A$.
- Train an SVM classifier for category c using the kernel:

$$k_{Lc}(\vec{x}_i, \vec{x}_j) = \exp\left(-\|A\vec{x}_i - A\vec{x}_j\|_2^2\right)$$

end for
- Use C SVM classifiers in one-versus-all way to classify $\vec{x}_0$ using the pool of kernels $\{k_{Lc}\}_{c=1}^C$.

3 Experimental Results

In this section, we will compare the performance of our proposed globally adaptive SVM algorithms with other nearest neighbor classifiers and standard SVM formulations on various UCIML, face and digit databases.

3.1 Faces, USPS and Isolet Database

We have used 5 face databases (yalefaces, AT&T, yalefacesB, caltechfaces and caltechfacesB), USPS digit database and Isolet database. The images

Figure 3: Example images from caltechfaces and caltechfacesB

in all databases are pre-processed for efficiency. That is, the dimensionality of the feature-vector representing each image is reduced by using Principal Component Analysis (PCA) method. The number of training images per category, number of testing images per category and the number of Eigen-vectors used for each database is given in table 1.

The classification performance in terms of the correctness rate of each of the following methods for faces, USPS and Isolet database is shown in figure 4 and 5:

- **KNN:** Simple 1-nearest-neighbor classification with the Euclidean distance.
- **SVM:** Standard SVM formulation, that is, a multi-class SVM with the Gaussian kernel is used. The $\mathcal{C}$ parameter is tuned through cross-validataion (that is searched from the set: $\{1, 10, 100, 1000\}$). The value of σ is set to be the average distance of k-nearest neighbors. A one-versus-all strategy is employed for multi-class classification. The results are labeled as 'SVM' in the performance graphs.
- **MEGM-KNN:** 1-nearest neighbor classifier with a distance metric learned from MEGM metric learning algorithm. The results are labeled as 'MEGM-KNN in the performance graphs.
- **GASVM1:** Algorithm 1.
- **GASVM2:** Algorithm 2.

The value of $\mathcal{C}$ for GASVM1 and GASVM2 algorithms is not optimized and set equal to 10 in all experiments. It can be seen from the figure 4 and 5 that GASVM methods performed better than other competing methods on all except caltechfacesB database (where MEGM-KNN performed best). Also, both GASVM formulations

Database	#Data	#Features	#Classes	PCA	#Train/Class	#Test/Class
yalefaces	165	77760	15	50	4	7
yalefacesB	5850	307200	10	20	10	20
caltechfaces	435	47500	29	30	5	10
caltechfacesB	435	47500	2	30	50	100
AT&Tfaces	400	10304	40	150	5	5
USPS	9298	256	10	50	100	30
Isolet	6238	617	26	172	50	30

Table 1: Details of databases used for classification.

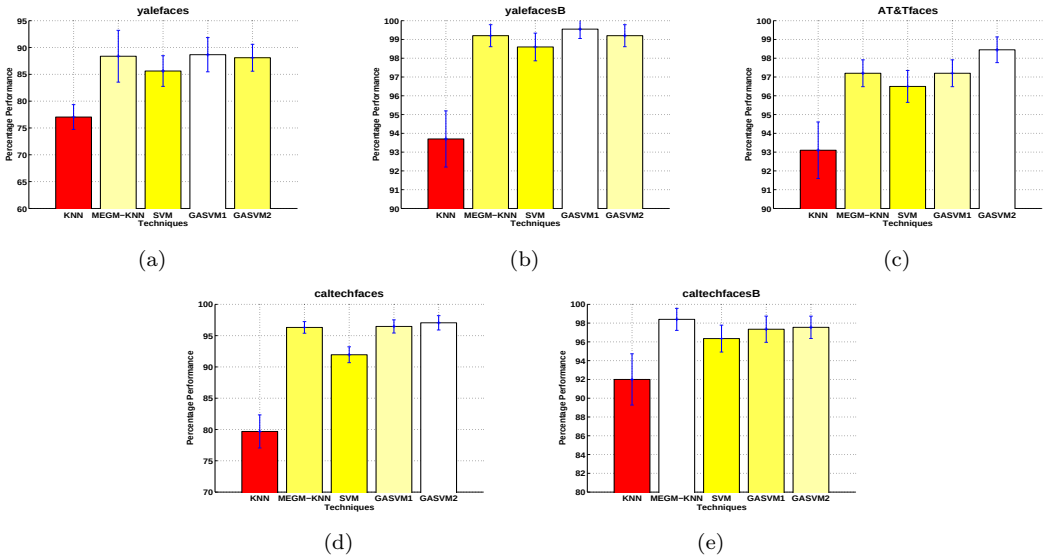


Figure 4: Comparison of the correctness rate of GASVM methods with KNN, MEGM-KNN and SVM methods on various face databases. The mean and standard deviation of the correctness rate over 10 runs (each run with different training data) is reported. Continued in figure 5.

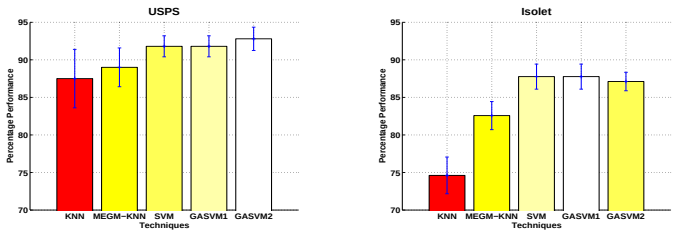
resulted in significant improvement in the classification performance of standard SVM classifier except for USPS database.

3.2 UCIML Repository Databases

The performance of GASVM algorithms is also evaluated on various UCIML databases. The error rate of each method is obtained using 40 rounds of 2 fold cross-validation. Prior to training, features in all the databases were normalized to have zero mean and a unit variance. The classification performance in terms of the error rate of KNN, SVM, OSVM, GASVM1 and GASVM2 for different databases is shown in figure 6. The details of KNN, SVM, GASVM1 and GASVM2 are the same as described in section 3.1. The details of OSVM method is given in the following:

- **OSVM:** Similar to the standard SVM, but both $\mathcal{C}$ and σ parameters are optimized using cross validation (we have called this formulation 'optimized SVM (OSVM)'). The $\mathcal{C}$ and σ parameters are searched from the sets: $\mathcal{C} = \{1, 10, 100, 1000\}$ and $\sigma = \{0.1, 0.5, 1, 2, 3, 5\}$ respectively. The results are labeled as 'OSVM' in the performance graphs.

It can be seen from figure 6, out of 21 UCIML databases, GASVM1 performed best on 11 of them. On the other hand, GASVM2 performed best on 4 databases. Both OSVM and KNN methods performed best on 3 databases each. It can be seen that the performance of GASVM1 is very close to standard and optimized SVM in most case. These results are encouraging because as mentioned before that SVM and OSVM requires tuning kernel parameters through an expensive cross-validation procedure, whereas GASVM methods have no such tweaking involved. It should be noted that the results obtained with GASVM2 have particularly high variance on most databases. For example, *balance* and *hayesroth* (figure 6(a), 6(c)). One likely reason for such high variance in the case of GASVM2 may have to do with its reliance on MEGM metric learning algorithm for learning the kernel parameters. Since, GASVM2 learns a different kernel for each category using MEGM algorithm, there are more



(a) (b)

Figure 5: Continued from figure 4. Results on USPS and Isolet database.

chances that local minima will affect its performance. Although GASVM methods performed well on a variety of methods, it can be seen from figure 6 that on *ionosphere*, *monks2* and *satimage* databases, the optimized SVM (OSVM) outperformed GASVM methods. Also surprisingly, KNN method performed better than standard SVM, OSVM and GASVM methods on *pageblock*, *parkinson* and *vowel* databases. As we discussed in section 2 that the formulation of GASVM assumes a Gaussian kernel. The success of GASVM methods on 15 out of 21 UCIML databases, suggests that the assumption of a Gaussian kernel is safe and reasonable on most databases. However, there are some exceptions depicted by some of the results in figure 6 for example figure 6(s), 6(t) and 6(u). The likely reasons for poor performance of SVM methods on these databases might be the actual kernel selection. An SVM trained with linear or polynomial kernel may have performed better.

4 Conclusion

In this paper we studied the effects of learning a data dependent distance metric on SVM classification. We proposed two algorithms GASVM1 and GASVM2 which are novel formulations of adopting a distance metric learning algorithm (MEGM) to train an SVM classifier. Empirical results on major UCIML, face and digit databases revealed that GASVM methods result in improving the classification performance of standard SVM classifier in most cases. Also results are comparable to SVM where both $\mathcal{C}$ and σ parameters are optimized. As metric learning algorithms in k-nearest neighbor settings have not been systematically studied in the context of kernel based methods for example SVM and Gaussian process classifiers, our results are encouraging and points to an interesting direction of further research.

References

[1] B. Scholkopf and A. Smola, Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond. The MIT Press, 2004.

[2] O. Chapelle, V. Vladimir, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," JMLR, 2002.

[3] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in ICCV, 2007.

[4] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," in CVPR, 2006.

[5] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer Series in Statistics, 2001.

[6] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighborhood component analysis," in NIPS, 2005.

[7] J. Davis and I. Dhillon, "Structured metric learning for high dimensional problems," in ACM SIGKDD conference on Knowledge Discovery and Data Mining, 2008.

[8] K. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in NIPS, 2006.

[9] B. Sriperumbudar, O. Lang, and G. Lanckriet, "Metric embedding for kernel classification rules," in ICML, 2008.

[10] J. Friedman, "Flexible metric nearest neighbor classification," Dept. of Statistics, Stanford University, Tech. Rep., 1994.

[11] N. Zaidi, D. M. Squire, and D. Suter, "A gradient-based metric learning algorithm for k-nn classifiers," in Proceedings of the Australasian Joint Conference on Artificial Intelligence, 2010.

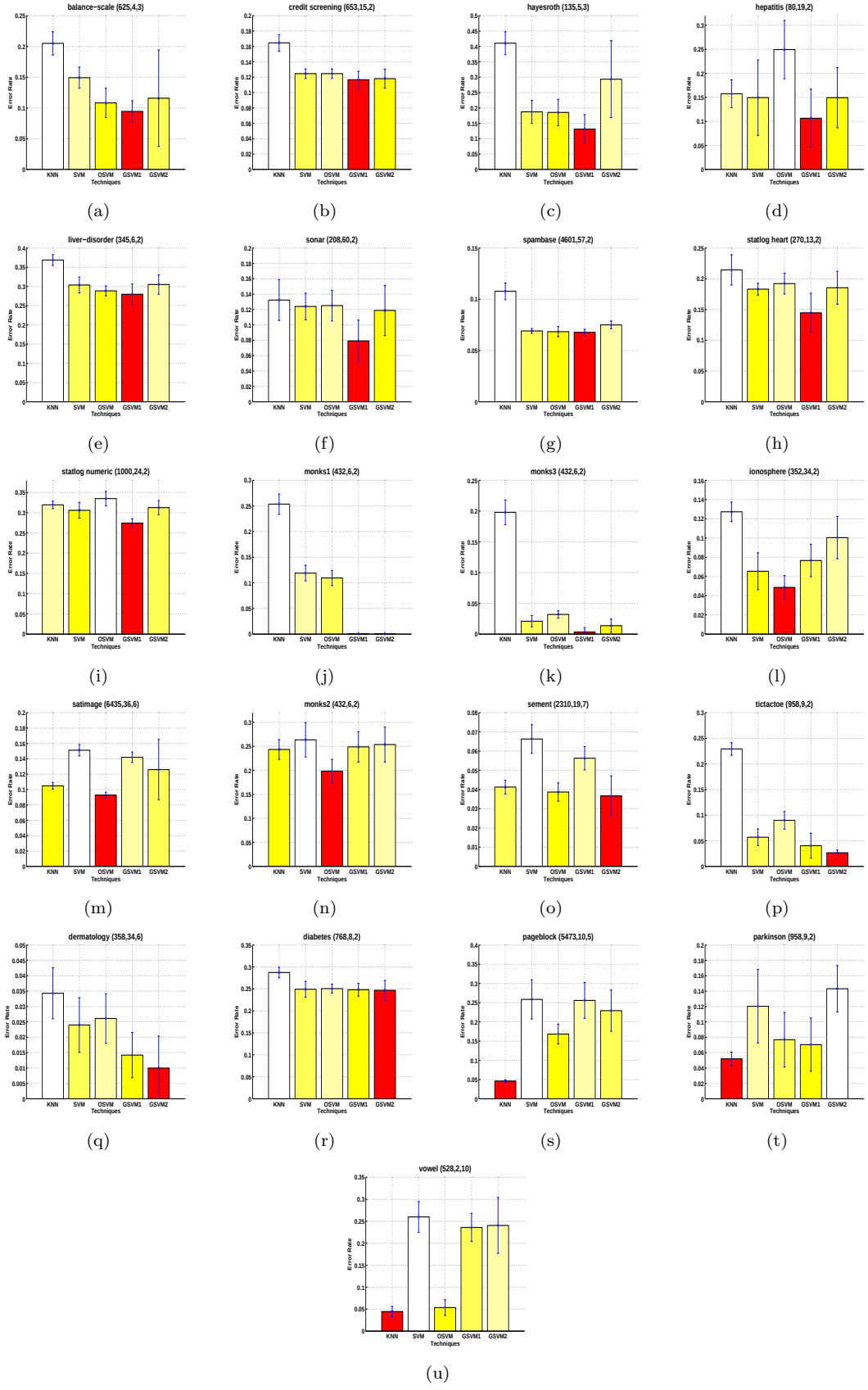


Figure 6: Comparison of the error rates of GASVM methods with KNN, SVM, OSVM on various UCIML databases. The number of data, features and classes for each database is shown in the title.