Local Adaptive Support Vector Machine for Object Recognition

Nayyar A. Zaidi David McG. Squire

Clayton School of Information Technology, Monash University, Melbourne VIC 3800, Australia

December 1, 2010

Support Vector Machine

- The Support Vector Machine (SVM) is an effective classification tool. Though extremely effective, SVMs are not a panacea. SVM training and testing is computationally expensive. Also, tuning the kernel parameters is a complicated procedure.
- On the other hand, the Nearest Neighbor (KNN) classifier is computationally efficient.
- In order to achieve the classification efficiency of an SVM and the computational efficiency of a KNN classifier, it has been shown previously that, rather than training a single global SVM, a separate SVM can be trained for the neighbourhood of each query point.

Research Goals

In this work, we have extended this Local SVM (LSVM) formulation. Our Local Adaptive SVM (LASVM) formulation trains a local SVM in a modified neighborhood space of a query point. The main contributions of the paper are twofold:

- First, we present a novel LASVM algorithm to train a local SVM.
- Second, we discuss in detail the motivations behind the LSVM and LASVM formulations and its possible impacts on tuning the kernel parameters of an SVM.

We found that training an SVM in a local adaptive neighborhood can result in significant classification performance gain. Experiments have been conducted on a selection of the UCIML, face, object, and digit databases.

Support Vector Machine (Contd)

Given a training set $S = \{(x_1, y_1), ...(x_n, y_n)\} \in \mathrm{R}^n \times \{-1, 1\}$, the decision function is found by solving the following convex optimization problem:

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

where α are the Lagrange multipliers, C controls the misclassification penalty and $k(.,.)$ is the kernel function.

Metric Learning and SVM

- Recently a lot of work has been done in the area of metric learning to improve the performance of k-Nearest Neighbor (k-NN) classifier.
- Metric learning algorithms aim at finding a metric that results in small intra-class and large inter-class distance.
- A 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.

Metric Learning and SVM (Contd)

- The kernel in equation 1 can be written as:

$$k(\vec{x_i}, \vec{x_j}) = \exp\left(\frac{-d^2(\vec{x_i}, \vec{x_j})}{2\sigma^2}\right) \tag{2}$$

  Applying metric learning, $d^2(\vec{x_i}, \vec{x_j})$ in equation 2 can be replaced by a more general metric based on a matrix L: i.e. $d_L^2(\vec{x_i}, \vec{x_j}) = (A\vec{x_i} - A\vec{x_j})^T(A\vec{x_i} - A\vec{x_j})$, where $L = A^T A$. We optimize A rather than L, as optimiz L entails a semi-positive constraint that is expensive to maintain.

- The kernel in equation 2 can thus be written as:

$$k(\vec{x_i}, \vec{x_j}) = \exp\left(-\|A\vec{x_i} - A\vec{x_j}\|_2^2\right) \tag{3}$$

- Using a learnt data-dependent distance metric can improve SVM classification performance.

Why Local Support Vector Machine (LSVM)?

- A major motivation for using LSVM is the gain in classification performance — yet while LSVM results in performance gain in some databases, performance deteriorates in many databases.
- LSVM, however, has an advantage over standard SVM as it involves no initial training and extension to greater numbers of classes is more feasible.
- The testing time is likely to increase, as an SVM classifier has to be trained for every query point. Despite this, in most databases having M classes, a query point is surrounded by instances of only m classes, and usually $m \ll M$. This can expedite the recognition phase rather than slowing it down.

Why local support vector machine (Contd)?

- Another motivation for training SVMs locally has to do with kernel's multiple scaling parameters. It is hoped that, at least locally, class conditional probabilities vary similarly across all features and using an isotropic kernel may not hurt much. This avoids the need to tune multiple kernel parameters in the SVM.

Local Adaptive Support Vector Machine (LASVM)

- The performance of LSVM is dependent on the neighborhood size in which it is trained. One expects that LSVM performance will be better for a larger neighborhood size, but increasing neighborhood size will result in a decrease in computational efficiency. Therefore, a neighborhood size has to be tuned such that an suitable trade-off is achieved between computational and classification efficiency.
- An alternative approach to increase in classification performance, while keeping the size of the neighborhood small, is to train a local SVM in an *adaptive* neighborhood of the query point. We call this formulation the Local Adaptive Support Vector Machine (LASVM).

LSVM

For any query point x_0, our local version of SVM (LSVM) works in the following way:

- Find the K nearest neighbors of x_0. If $N = \text{cardinality of training dataset}$ then, $K = k \times \log2(N)$, typically $k \in \{1, 2, 3, 5, 10\}$.
- If the labels of all K points are same, x_0 belongs to the corresponding class and the procedure exits.
- If the labels are different, a one-versus-all SVM is trained for each class present, and x_0 is labeled accordingly.

Mean Square Error based Metric Learning Algorithm (MEGM)

- In this work, we have used the MEGM metric learning algorithm from our previous work. Other metric learning algorithms can also be used. Notable examples include Neighborhood Component Analysis (NCA), Information Theoretic Metric Learning (ITML), etc.
- A lot of work has also been done in locally adaptive metric learning algorithms. A local SVM can be trained on the transformed neighborhood that results from a local adaptive metric.

Algorithm 1 LASVM: Train an SVM classifier in the adapted neighborhood of a query point.

Require:

- Testing data: x_0
- Training data: $\{x_n, y_n\}_{n=1}^{N}$, where x is a p dimensional feature vector, N is the number of training data, y is training label $y = \{1, 2, ...M\}$, where M is the number of classes.
- $K = k \times \log2(N)$, typically $k = [1, 2, 3, 5, 10]$.

- Learn a data-dependent distance matrix L such that $L = A^T A$ using the MEGM metric learning algorithm.
- Transform both training and testing data using matrix A.

- Apply the LSVM procedure on the transformed training and testing data.

Experimental Results

The performance of LSVM and LASVM is compared on following databases:

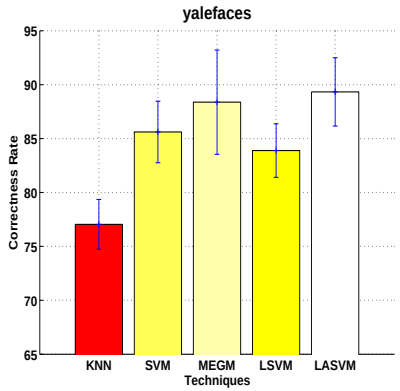
- yalefaces, yalefacesB, AT&Tfaces
- caltechfaces, caltechfacesB
- isolet, USPS, Coil100
- UCIML databases

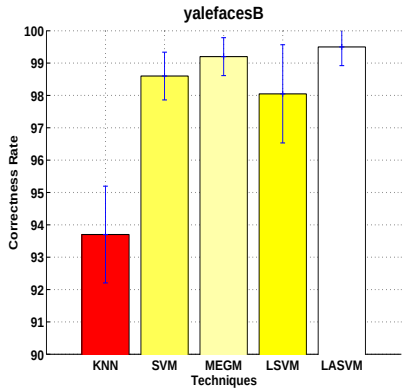
Experimental Results

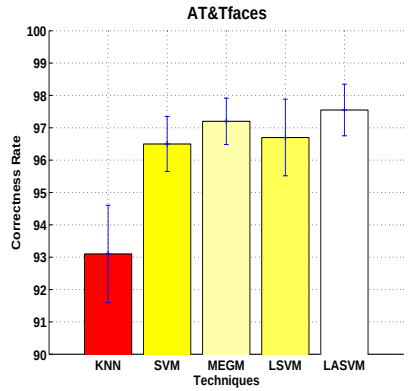
- **KNN:** 1 Nearest neighbor classification.
- **SVM:** The C parameter for the SVM is tuned through cross-validataion, searched from the set: $\{1, 10, 100, 1000\}$. The value of σ is set to the average distance of k nearest neighbor ('standard SVM') [1] .
- **MEGM:** 1 Nearest neighbor classification in the transformed space. The transformation is parameterized by matrix learned using MEGM algorithm.
- **OSVM:** In the case of the UCIML databases, for a better comparison of LSVM and LASVM with standard SVM, both C and σ parameters are optimized for SVM using cross validation ('optimized SVM (OSVM)'). An SVM giving the best performance from $C = \{1, 10, 100, 1000\}$ and $\sigma = \{0.1, 0.5, 1, 2, 3, 5\}$ is chosen.

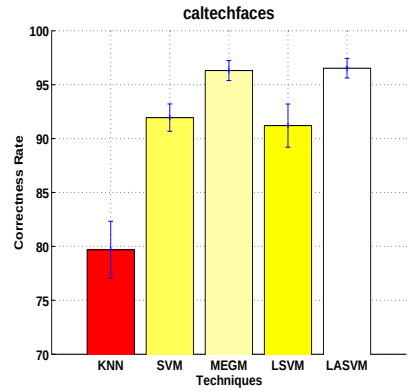
[1] note that all SVM formulations are trained with a Gaussian kernel and a one-versus-all strategy is employed.

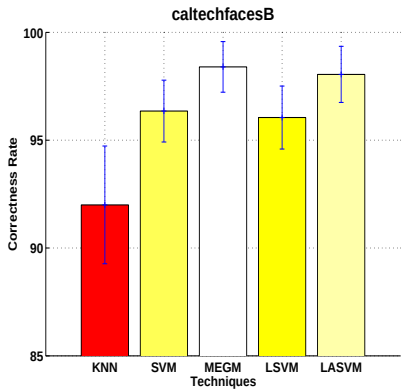
Results

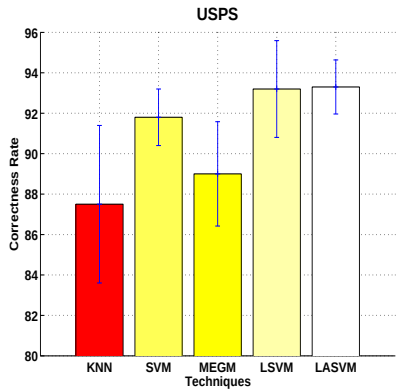

yalefaces
Correctness Rate
95
90
85
80
75
70
65
KNN
SVM
MEGM
LSVM
LASVM
Techniques

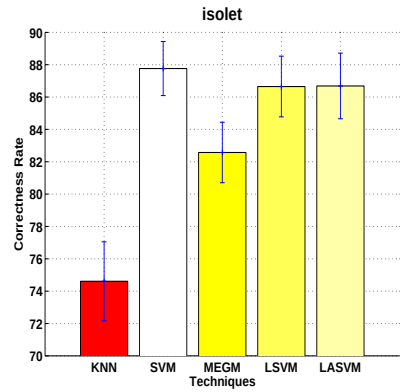

yalefacesB
Correctness Rate
100
99
98
97
96
95
94
93
92
91
90
KNN
SVM
MEGM
LSVM
LASVM
Techniques

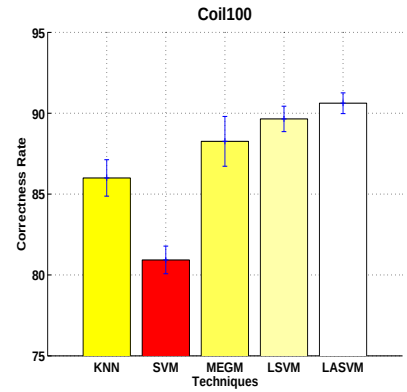

AT&Tfaces
Correctness Rate
100
99
98
97
96
95
94
93
92
91
90
KNN
SVM
MEGM
LSVM
LASVM
Techniques


caltechfaces
Correctness Rate
100
95
90
85
80
75
70
KNN
SVM
MEGM
LSVM
LASVM
Techniques

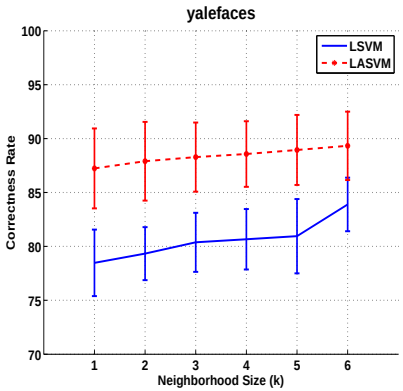

caltechfacesB
Correctness Rate
100
95
90
85
KNN
SVM
MEGM
LSVM
LASVM
Techniques

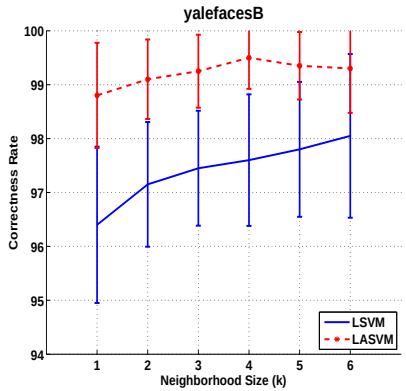

USPS
Correctness Rate
96
94
92
90
88
86
84
82
80
KNN
SVM
MEGM
LSVM
LASVM
Techniques

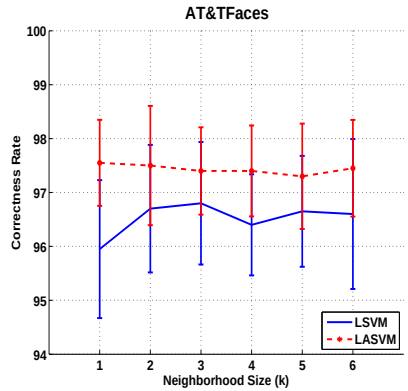

isolet
Correctness Rate
90
88
86
84
82
80
78
76
74
72
70
KNN
SVM
MEGM
LSVM
LASVM
Techniques

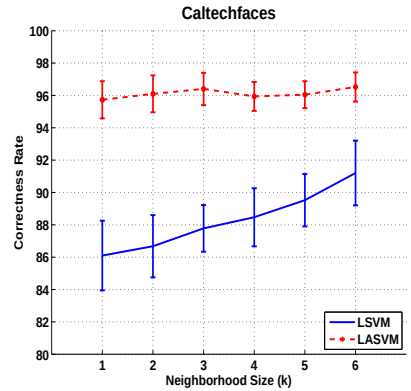

Coil100
Correctness Rate
95
90
85
80
75
KNN
SVM
MEGM
LSVM
LASVM
Techniques

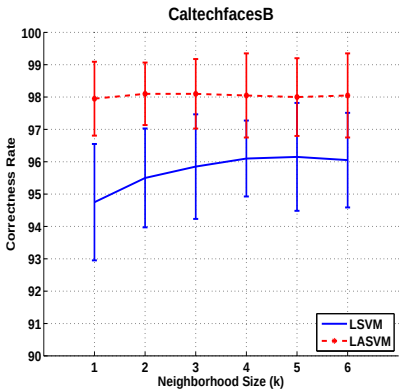
Results

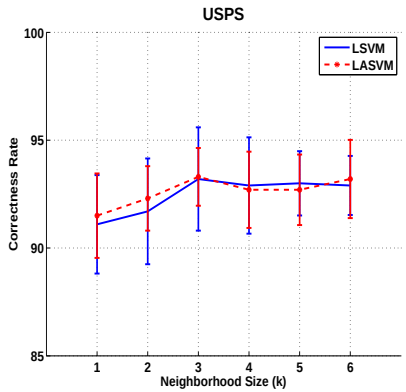

yalefaces
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

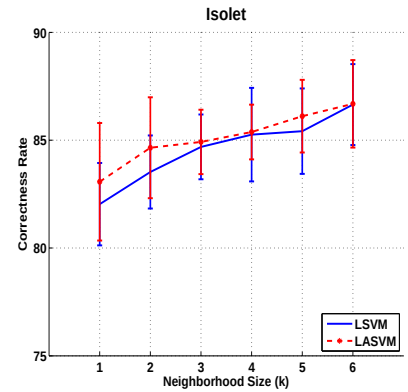

yalefacesB
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

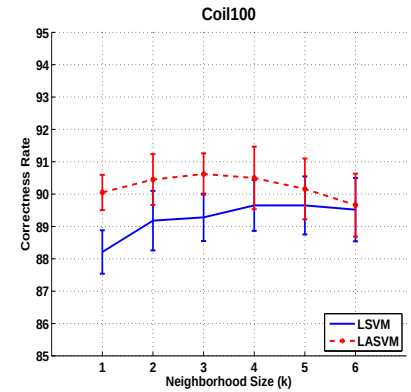

AT&TFaces
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)


Caltechfaces
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

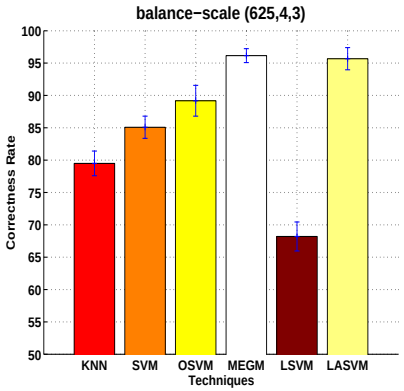

CaltechfacesB
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

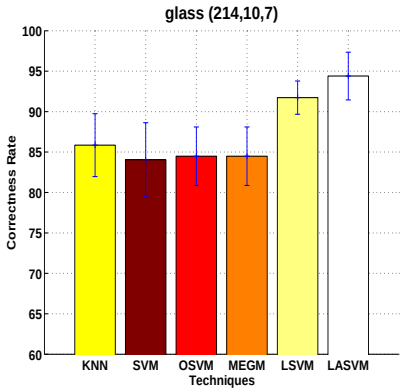

USPS
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

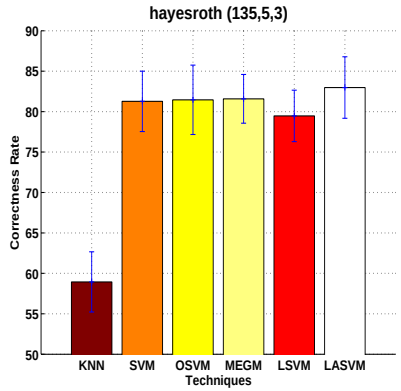

Isolet
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

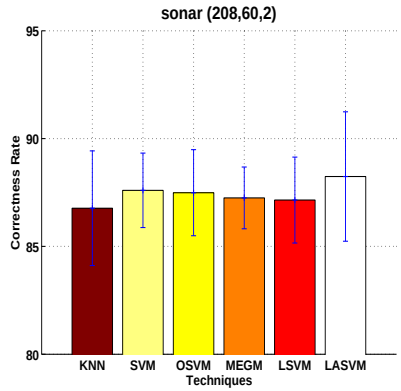

Coil100
LSVM
LASVM
Correctness Rate
Neighborhood Size (k)

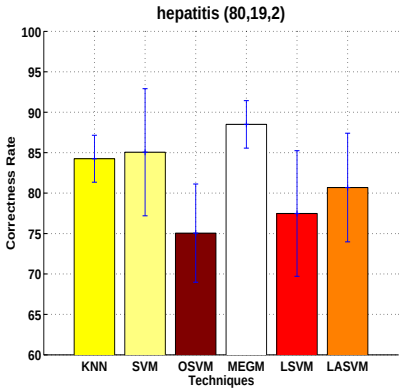
Results (UCIML databases)

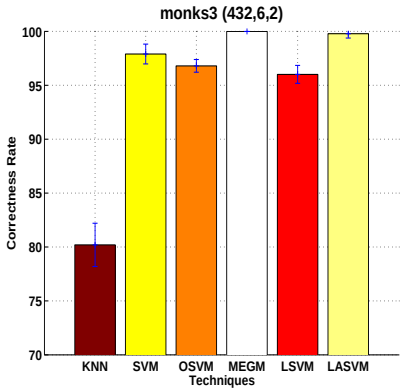


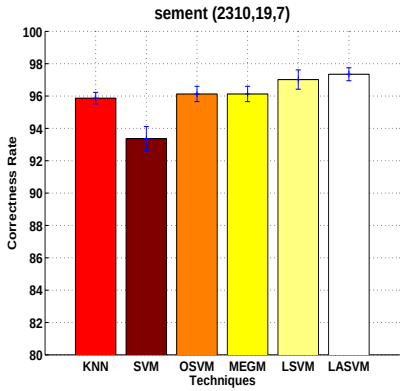


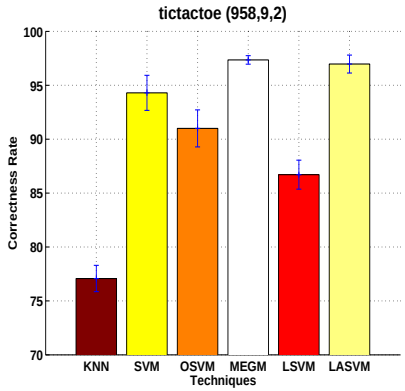




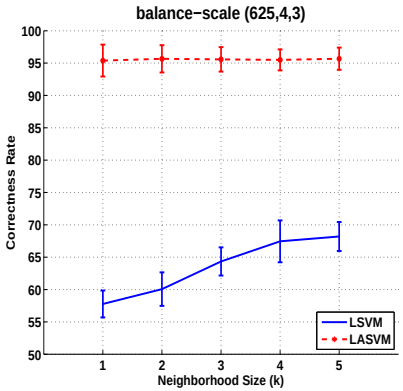


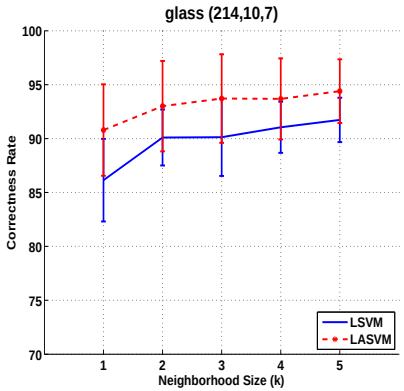


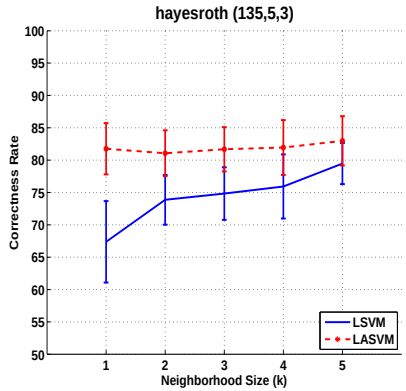


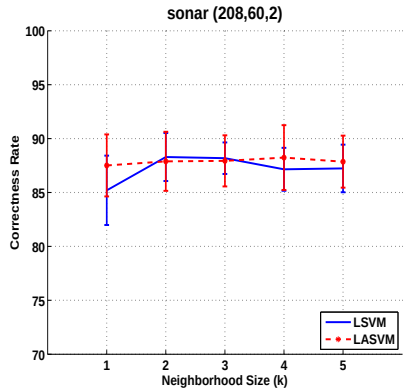


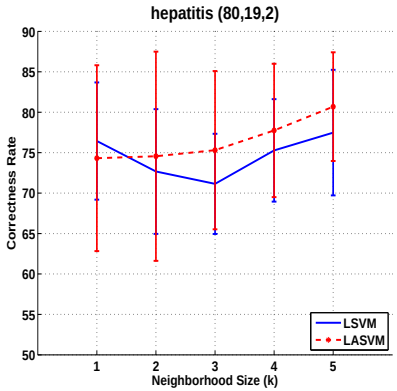
Results (UCIML databases)

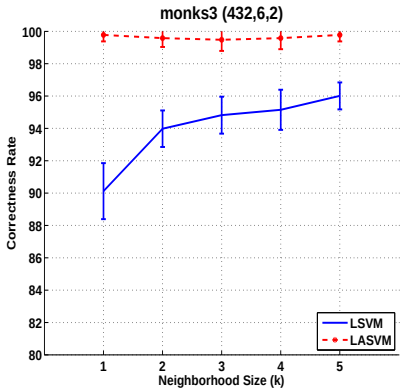


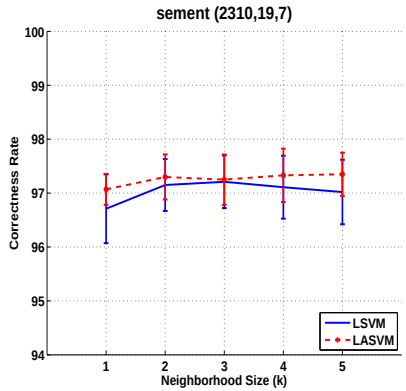


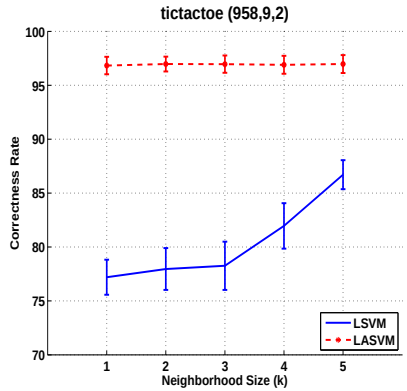












Discussion

- First, no SVM parameters are optimized in the cases of LSVM and LASVM. Our results are thus encouraging not only for LASVM, but also for LSVM, as LSVM performance is comparable in some cases to standard SVM and OSVM. For example, in the case of Coil100 database, using LSVM results in a very large performance gain over standard SVM.
- Secondly, due to space constraints, we have not reported computational time in this work. In our experiments we found that training LSVM and LASVM was far more efficient in term of computational efficiency compared to standard SVM, provided we keep the neighborhood size reasonably small.

Discussion (cont'd)

- Thirdly, although LSVM performs better than SVM in some cases, it does worse in many others. This suggests that the LSVM algorithm proposed in Zhang et al., ought not be used with simple Euclidean distance. Indeeed Zhang et al. have proposed to use LSVM with a specifically designed distance measure. Our locally adaptive formulation LASVM resulted in far superior performance when compared with LSVM.

Conclusion

- LASVM resulted in significant improvement over the performance of not only LSVM, but also SVM, KNN, and MEGM classifiers on faces, object, digit, and UCIML databases.
- We found that, unlike LSVM, LASVM performance is not greatly affected by the neighborhood size. These results are promising and point to an interesting direction for further research.
- We also highlighted some advantages of LSVM methods and described how they can help alleviate kernel parameter tuning problems.

Conclusion (cont'd)

- Our results on LSVM suggest that it should not be viewed as a replacement for SVM, but as a compromise between SVM and KNN. LSVM is especially useful in cases where the number of classes is very large, as LSVM is faster than standard SVM and has better performance than KNN.
- Our LASVM results, on the other hand, are encouraging and needs to be investigated further with other metric learning algorithms.

Questions