Chapter 6
Geometric Incremental Support Vector Machine for Object Detection from Capsule Endoscopy Videos

Xiaohui Yuan and Mohamed Abouelenien
University of North Texas, Department of Computer Science and Engineering, Denton, TX, USA

Balathasan Giritharan
CityGrid Media, West Hollywood, CA, USA

Jianguo Liu
University of North Texas, Department of Mathematics, Denton, TX, USA

Xiaojing Yuan
University of Houston, Department of Engineering Technology, Houston, TX, USA

6.1 Introduction

Capsule endoscopy (CE) is a method used to visualize the entire small intestine. It is a widely adopted procedure for diagnosing gastrointestinal diseases including obscure bleeding, Crohn's disease, gastric ulcers, and colon cancer. The CE videos used in this research were produced with the Pillcam® by Given Imaging. The imaging component of this system is a vitamin-sized capsule that comprises a color CMOS camera, a battery, a light source, and a wireless transmitter. The device captures two images per second for approximately eight hours and generates approximately 55,000 color images with a size of 256 $\times$ 256 pixels during the life of its usage.

Reviewing CE videos to make diagnostic decisions is a tedious task and is achieved by watching the video playback and marking suspicious frames and anatomical landmarks. It usually takes more than one hour to annotate a full-length video, and a typical mid-size hospital produces an average of twelve CE videos per day.

Given the large amount of training data, computer algorithms are in great demand to reduce the review time by identifying frames that contain signs of lesion, bleeding, and polyps, as well as segment videos into gastrointestinal sections. Many existing learning algorithms require all training data to be present in memory to achieve the best generalization performance. Limited by the computing power and memory size, it is usually difficult to implement such a learning scheme. Incremental learning has great potential to accommodate the inclusion of examples that become available over time or represent a change of perception. The initial data set can be used to create a model; when new data becomes available, it is integrated to update the classifier. In practice, clinical videos are acquired over time. Furthermore, knowledge of the visual appearance of the diseases in CE video changes over time due to the relatively shorter practice time. It would be practical to build a classifier based on initial data and revise the classifier as new examples arrive.

A key question of incremental methods is how to retain knowledge from the training examples in each repetition to maximize the unbiased representation of underlying data distribution. Retaining some key examples, e.g., support vectors in a support vector machine (SVM), works well in cases where the existing examples closely represent the topography of the class boundary. However, if a new instance dramatically changes the topography and hence the decision hyperplane, some previously removed examples could become the margin mover.

This chapter presents an incremental learning method that extends the geometric SVM to multiclass classification with large training data. The proposed method identifies important examples and models the data such that when new examples become available, a classifier is built without revisiting all of the past data available but with generalization accuracies, which are comparable to those obtained in the batch-learning setting.

6.2 Related Work

6.2.1 Related work on CE video analysis for automatic object detection

Among efforts in computer-aided CE video analysis, color and texture features are used in many applications,[24] particularly for detecting heterogeneous objects, e.g., ulcers and polyps.[3,10,28,39] Many classification algorithms have been applied to video analysis including neural networks,[39] SVMs,[16] and thresholding. Despite improvements, many previous studies were evaluated with a small number of examples, and to the best of our knowledge no performance was reported with respect to the entire videos.

Table 6.1 Experimental data and detection outcomes. "–" indicates not reported in the paper. The sensitivity, specificity, and accuracy are in percentages.

Reported Studies	Data Set Size			Performance		
	Total	Abnormal	Normal	Sen.	Spe.	Acc.
Kodogiannis and Boulougoura[24]	140	35	35			95.7
Kodogiannis and Lygouras[25]	140	35	35			97.1
Vilarino et al.[39]	400	100	300			95.5
Coimbra and Cunha[10]	1000					87
Lau and Correia[26]	1705	577	1128	88.3		
Li and Meng[27]	60	30	30	65.2	82.5	
Li and Meng[28]	400	200	200	91	93	
Jung et al.[18]	2000	1000	1000	92.8	89.5	
Barbosa et al.[3]	204	100	104	98.7	96.6	
Karargyris and Bourbakis[19]		20	30	75	73.3	
Karargyris and Bourbakis[20]	50	10	40	100	67.5	

Table 6.1 summarizes the characteristics of experimental data sets and performances in recent related work on automatic detection using CE videos. Despite the use of different features and classification methods, the experimental data and performances vary greatly. Among these studies, results in eight studies were generated from experiments using 1000 examples or less. Two studies used a moderately larger number of examples. Compared to the number of frames available in a CE video (approximately 50,000), however, the training data set size is small. Ideally, if the training set is well selected and comprehensive, the classifier can achieve satisfactory generalization performance. It is unclear if the formed cohort represents the true data distribution. An important question awaits investigation: "Given the relatively small number of positive examples from CE videos, how does one train learning algorithms to achieve minimal false negative detections?"

6.2.2 Related work on incremental learning using SVMs

Although a large number of training examples helps reduce the generalization error, the learning process can become computationally expensive, if not infeasible. Efficient and scalable approaches are needed that can modify the knowledge structure in an incremental fashion without having to revisit all of the previously processed data.

Attempts at an incremental SVM started by retaining the support vectors. The method in Syed et al.[38] keeps only the support vectors at each incremental step. The model obtained via this strategy will be the same or very similar to what would have been obtained by using all training examples. Mitra et al.[32] used an error-driven technique in the incremental SVMs. In addition to the support vectors, this method keeps a number of non-support-vector examples. Given a trained $SVM^{(t)}$ at iteration t, the SVs of $SVM^{(t)}$ (along with a certain number of correctly classified and misclassified instances) are used to train the new model $SVM^{(t+1)}$. Alternatively, Domeniconi and Gunopulos[13] proposed a method that keeps only the misclassified examples. When a given number of misclassified examples is collected, the update occurs. The support vectors of the last-trained SVM, along with the misclassified instances, are used as training data to obtain the new model. The assumption of minimum change in the hyperplane serves as the foundation of the previous methods.

Katagiri and Abe[21] proposed using one-class SVMs to select support vectors, which reduces the possibility of support vectors being deleted when the hyperplane is rotated. A hypersphere is generated for each class, and only the instances lying close to the boundary of the hypersphere are retained as candidate support vectors for future updates. Although this method handles the rotation of the decision boundary, the assumption of a hypersphere to model data distribution is unrealistic in many real-world applications.

To manage the space complexity and size of the representative data set, Hernandez et al.[14] employed a multiresolution approach. Agarwal et al.[40] demonstrated that the concept of the span of support vectors can be used to build a classifier that performs reasonably well while satisfying space and time constraints, thus making it suitable for online learning. Mitra et al.[33] presented probabilistic SVMs wherein the training set is refined by active query from a pool of unlabeled data. Orabona et al.[34] proposed an online algorithm that approximately converges to the standard SVM solution each time new examples are added. This method uses a set of linearly independent observations and tries to project every new observation onto the set obtained so far, thus reducing time and space requirements at a negligible loss of accuracy. Proximal SVM[36] employs a greedy search across the training data to select the basis vectors of the classifier and tunes parameters automatically using the simultaneous perturbation stochastic approximation after incremental additions are made.

Instead of selecting training examples randomly, Chen et al.[9] divided the training set into groups using the *k*-means clustering algorithm. In active query, a weight is assigned to each example according to its confidence, which is calculated from the error upper bound of the SVM to indicate the closeness of the current hyperplane to the optimal one.

Another key issue in incremental learning is to adapt to the nonstationary underlying data distribution. Cauwenberghs and Poggio[6] developed an incremental and decremental SVM method that divides the training set into three categories: the margin SVs, the error SVs (ones that violate the margin but are not necessarily misclassified), and ignored vectors (ones within the margin). When a new instance is misclassified, the SVM is updated. Bookkeeping is used to categorize examples, the complexity of which is $O(n^3)$ for each incremental example. A later work of Diehl and Cauwenberghs[12] reduced the computational cost by using "leave-one-out" error estimation. Again, the methods assume that the hyperplane does not change significantly.

Klinkenberg and Joachims[23] proposed a method to handle drift in SVMs. The drift represents changes to the underlying distribution of the data collected over an extended period for learning tasks. The method maintains a window to the training data stream and adjusts its size so that the estimated generalization error is minimized. Shilton et al.[37] addressed the sequentially arriving data and parameter variation using a warm-start algorithm. It allows efficient retraining of a SVM after adding a small number of additional examples. Boubacar et al.[5] employed an online clustering algorithm that is developed to learn continuously evolving clusters from nonstationary data. This algorithm uses a fast incremental learning procedure to account for model changes over time. Dedicated to online clustering in multiclass environment, the algorithm is based on an unsupervised learning process with self-adaptive abilities.

6.3 Geometric Incremental Support Vector Machines

Geometric and quadratic optimization views of SVMs were shown to be equivalent.[4,11] A geometric SVM represents each class as a convex hull and finds the minimum distance between the two.[22] To address nonseparable classes, the reduced convex hull (RCH)[11] was developed.[30,31] The method of incremental learning presented here extends the RCH concept and proposes that convex skin represent key examples in training, as well as a means of finding convex skins.

6.3.1 Geometric support vector machines

Let x be a data point in a convex hull C. According to Caratheodory's theorem, x can be represented as a convex combination of a finite number of points in C:

$$x = \sum_{j\ \ 1}^{k} \lambda_j x_j, \text{ where } \lambda_j \geq 0, \text{ and} \sum_{j\ \ 1}^{k} \lambda_j = 1. \tag{6.1}$$

Given a set of data points X, the convex hull is a linear combination of all the elements in X and can be represented as follows:

$$C(X) = \left\{ \sum_{i\ \ 1}^{k} \alpha_j x_i;\ x_i \in X,\ 0 \leq \alpha_j \leq 1,\ \sum_{i\ \ 1}^{k} \alpha_i = 1 \right\}. \tag{6.2}$$

Reduced convex hull[4] (also known as soft convex hull[11]) is the set of all convex combinations of elements of X, denoted by $R(X, \mu | \mu < 1)$, as follows:

$$R(X, \mu) = \left\{ \sum_{i\ \ 1}^{k} \alpha_j x_i;\ x_i \in X,\ 0 \leq \alpha_i \leq \mu,\ \sum_{i\ \ 1}^{k} \alpha_i = 1 \right\}. \tag{6.3}$$

The difference between a convex hull and a RCH is that the weight factor α_i is bounded by μ in a RCH. Using a suitable μ for each class, two overlapping classes can be transformed into a linearly separable case.[4,11,31]

However, the RCH provides no means of finding the extreme points. To overcome this, a compressed convex hull was proposed.[35] It, however, makes explicit assumptions on the kernel, which limits its application.

Geometric SVM represents classes as convex hulls and solves the problem by finding the minimum distance.[22] Given a set of examples $X = \{x_1, x_2, ..., x_n\}$, the function ϕ maps each instance into a features space $\phi(x_i)$. For simplicity, ϕ_i is used here to denote $\phi(x_i)$, and the mapped examples form a feature set $\Phi = \{\phi_1, \phi_2, ..., \phi_n\}$. The convex hull $C(\Phi)$ is rewritten as follows:

$$C(\Phi) = \left\{ \sum\nolimits_{j\ \ 1}^{k} \alpha_i \phi_i | \phi_i \in \Phi, 0 \le \alpha_i \le 1, \sum\nolimits_{j\ \ 1}^{k} \alpha_i = 1 \right\}. \tag{6.4}$$

Similarly, a RCH is the set of convex combinations of instances in Φ with α_i bounded by μ as follows:

$$R(\Phi, \mu) = \left\{ \sum\nolimits_{i\ \ 1}^{k} \alpha_i \phi_i |\ \phi_j \in \Phi,\ 0 \le \alpha_i \le \mu, \sum\nolimits_{i\ \ 1}^{k} \alpha_i = 1 \right\}. \tag{6.5}$$

The decision boundary is then perpendicular to the nearest points between RCHs and can be found following Bennett's method.[4]

6.3.2 Geometric incremental support vector machine (GISVM)

Our method extends the concept of RCH and defines the skin of a convex hull. The idea is that only the examples within the skin are most informative and should be retained for future training, which is similar to Katagiri's idea,[21] but a model for the data distribution is not specified. When additional examples become available, they are used to update the SVM together with the skin of the current convex hull. In such a way, many fewer instances are used in a training process. In addition, with a superset of the possible SVs retained, missing SVs due to significant changes to the data distribution caused by the addition of new examples is avoided.

The skin of a convex hull consists of the outer-most vertices (i.e., examples). Given bounding factors μ_u and μ_l, $0 \le \mu_l < \mu_u \le 1$, the skin $S(\Phi, \mu_l, \mu_u)$ of a convex hull $C(\Phi)$ consists of instances between two RCHs and can be expressed as follows:

$$S(\Phi, \mu_l, \mu_u) = \{\phi_i | \phi_i \in \{R(\Phi, \mu_u) \quad R(\Phi, \mu_l)\}\}. \tag{6.6}$$

When the data set is dense enough and evenly distributed in the space, the geometric center can be used to find the extreme points of the convex hull. However, this is usually not the case in real-world applications. Due to the lack of knowledge of data distribution, the above procedure could miss less-prominent extreme points. Thus, a recursive method is proposed that finds the vertices (i.e., extreme points) of a convex hull to represent the skin.

It is said that $\phi_j \in \Phi$ is an extreme point of convex hull $C(\Phi)$ if there exists a direction d in terms of two instances, i.e., $d = \phi_b \quad \phi_a$, and $\phi_a,\ \phi_b \in C(\Phi)$, such that

$$\phi_j = \max{}_{\phi k \in \Phi}(\phi_k \quad \phi_a, d), \tag{6.7}$$

where $(\phi_k \quad \phi_a, d)$ is the inner product of the difference vectors with respect to ϕ_a and the direction d.

The extreme points are found in two steps: first, a set of initial extreme points are identified based on the center of gravity; and second, additional extreme points are then found via recursively searching along the direction defined by a pair of extreme points.

For a set of feature vectors Φ, the gravity center $\overline{\Phi}$ is approximated with the arithmetic average, i.e., $\overline{\Phi} = \sum_{i\ \ 1}^{n} \frac{1}{n}\phi_i$. The initial set of extreme points is identified by projecting each point $\phi_j \in \Phi$ to the direction $d(\phi_m) = \phi_m \quad \overline{\Phi}$ and selecting the ones that give the maximum projection magnitude:

$$E_{\text{seed}}(X) = \{\phi_n\ |\arg\max{}_{\phi n} P(\phi_n,\ d(\phi_m)), \forall\ \phi_m, \phi_n \in \Phi\}, \tag{6.8}$$

where $P(\phi_n,\ d(\phi_m))$ denotes the projection of ϕ_n to $d(\phi_m)$.

The explicit expression of the feature vectors ϕ_i is not needed to compute the extreme points in the above procedure. The projection $P(\phi_n, d(\phi_m))$ in the feature space can be achieved by the kernel operation in the input space as follows. Given two feature vectors ϕ_a and ϕ_b in Φ, the projection of vector ϕ_c is $P(\phi_c, d(\phi_a, \phi_b))$. Thus,

$$\begin{aligned} P(\phi_c, d(\phi_a, \phi_b)) &= \langle \phi_b \quad \phi_a, \phi_c \quad \phi_a \rangle \\ &= \langle \phi_b, \phi_c \rangle \quad \langle \phi_b, \phi_a \rangle \quad \langle \phi_a, \phi_c \rangle + \langle \phi_a, \phi_a \rangle \\ &= \sum_i b_i \phi_i \cdot \sum_j c_j \phi_j \quad \sum_i b_i \phi_i \cdot \sum_j a_j \phi_j \\ &\qquad \sum_i a_i \phi_i \cdot \sum_j c_j \phi_j \quad \sum_i a_i \phi_i \cdot \sum_i a_i \phi_i \\ &= \sum_i \sum_j b_i c_j K(x_b, x_c) \quad \sum_i \sum_j b_i a_j\ K(x_b, x_a) \\ &\qquad \sum_i \sum_j a_i c_j\ K(x_a, x_c) \quad \sum_i \sum_i a_i a_i\ K(x_a, x_a), \end{aligned} \tag{6.9}$$

where $\sum\limits_i a_i \phi_i$, $\sum\limits_i b_i \phi_i$, and $\sum\limits_i c_i \phi_i$ are convex representations of feature vectors ϕ_a, ϕ_b, and ϕ_c, respectively. For a vector $\phi_j \in \Phi$, its coefficient vector equals $[0,\ 0,\ \ldots,\ 1,\ \ldots,\ 0,\ 0]'$, within which the index value of the number 1 is j. For a vector $\phi_k \in C\ \Phi$ but $\phi_k \in C(\Phi)$, the values in its coefficient vector are in the range of [0, 1), e.g., the coefficient vector of the gravity center $\overline{\Phi}$ is $\left[\frac{1}{n},\ \frac{1}{n},\ \ldots,\ \frac{1}{n}\right]'$.

An example is illustrated in Fig. 6.1(a). The solid squares denote the examples, and the gravity center is marked with a large circle. The projected vectors are marked with solid dots. Using the proposed method, three extreme

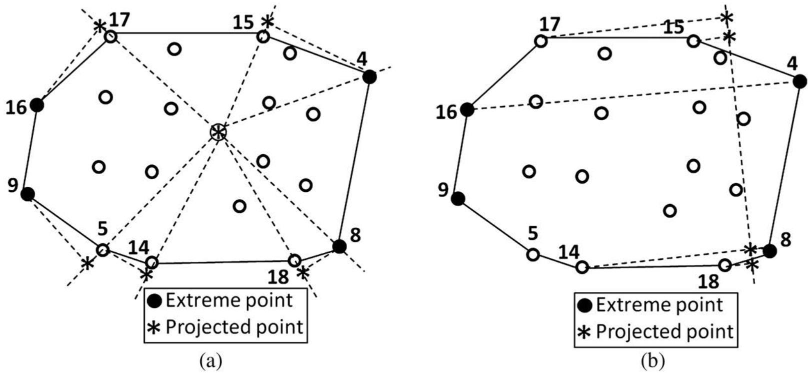

Figure 6.1 Finding extreme (a) data and (b) seed points recursively.

points are identified and highlighted with solid squares. For example, point 16 is identified as an extreme point because it gives the greatest projection to $d(x_{16}, \overline{X})$ [as well as $d(x_{15}, \overline{X})$]. However, instances 14, 15, 17, and 18 are extreme points that are missed by the process.

The primary cause of missing extreme points is the insufficient number of examples, which could be exaggerated in high-dimensional cases. If data points in the feature space are known, classical algorithms such as QuickHull[2] and Gift Wrapping[17] can be used to complete the search. The idea of the proposed algorithm is to recursively search along the perpendicular directions of the convex hull boundaries, which is presented in Algorithms 1 and 2.

Algorithm 1: Search for extreme points.

```
Require: $\Phi$ and $E$

1. Randomly select $\phi_p,\ \phi_q \in E$
2. Randomly select $\phi_m \in \Phi$ and $m \neq p, m \neq q$
3. Identify probing direction $d^*$ using Eq. (6.10)
4. $\Phi \quad \leftarrow \{\phi_i | P(\phi_i, d^*) < 0\}$
5. $\Phi_+ \leftarrow \{\phi_i | P(\phi_i, d^*) > 0\}$
6. $E \leftarrow E \cup Probing(\Phi_+, d^*, \phi_p, \phi_q)$
7. $E \leftarrow E \cup Probing(\Phi \quad , \quad d^*, \phi_p, \phi_q)$
8. Return $E$
```

This algorithm randomly selects two extreme points $\phi_p, \phi_q \in E$ and another instance $\phi_m \in \Phi$. The searching direction d^* can then be determined as follows:

$$d^* = \phi_m \quad \phi_p \quad P\left(\phi_m, d(\phi_p, \phi_q)\right) \frac{\phi_q \quad \phi_p}{||\phi_q \quad \phi_p||}. \tag{6.10}$$

A hyperplane through $\phi_q \quad \phi_p$ and perpendicular to d splits the space into two halves. The projections of instances, i.e., $P(\phi_i, d^*)$, that are on the same sides as ϕ_m are positive, denoted by Φ_+; whereas the projections of the rest instances are negative, denoted by Φ . Hence, the further searching for extreme points is divided into two parts, as shown in Algorithm 1.

Searching in each half space is achieved recursively using a pair of identified extreme points ϕ_p and ϕ_q. Let Φ' denote the instances in the half space. With a random instance ϕ_m in Φ', a probing direction d^* can be determined by Eq. (6.10) that points toward the outside of the convex hull; otherwise, change its direction. Hence, an extreme point is identified in Φ' following Eq. (6.7). ϕ_m is paired with ϕ_p and ϕ_q to split the feature space for further probing. The process stops when no additional points exist in Φ'.

Algorithm 2: Recursively probe and search for the extreme points Probing (Φ', d, ϕ_p, ϕ_q).

```
Require: $\Phi' \subseteq \Phi$, $d$, $\phi_p$, and $\phi_q$
 1. $F \leftarrow \varnothing$
 2. Randomly select $\phi_m \in \Phi'$ and $m \neq p,\ m \neq q$
 3. If $\Phi' \neq \varnothing$, then
 4. Identify probing direction $d^*$ using Eq. (6.10)
 5. If $\langle d^*, d \rangle < 0$, then
 6. $d^* \leftarrow \quad d^*$
 7. End if
 8. $d^* \leftarrow \frac{d^*}{||d^*||}$
 9. $F \leftarrow F \cup \{\phi_e | \phi_e = \arg\max_{\phi K \in \Phi'} P(\Phi_k, d^*)\}$
10. For all $\phi_i \in \Phi'$, do
11. If $P(\phi_i,\ d^*) > 0$, then
12. $\Phi'' \leftarrow \Phi'' \cup x_i$
13. End if
14. End for
15. $F \leftarrow F \cup Probing(\Phi'', d, \phi_p, \phi_e)$
16. $F \leftarrow F \cup Probing(\Phi'', d, \phi_q, \phi_e)$
17. End if
18. Return $F$
```

Figure 6.1(b) illustrates an example of probing in a half space. The dotted lines depict the projections of the instances. The two extreme points are 17 and 18, which determine the probing direction (d and –d in Algorithm 1). Extreme points 1 and 12 are found.

In the algorithm, the magnitude of vectors $\phi^{(2)} \quad \phi^{(1)}$ is calculated as follows:

$$\begin{aligned}\|\phi^{(2)} \quad \phi^{(1)}\| &= \left[\langle\phi^{(1)}, \phi^{(1)}\rangle + \langle\phi^{(2)}, \phi^{(2)}\rangle \quad 2\langle\phi^{(2)}, \phi^{(1)}\rangle\right]^{\frac{1}{2}} \\ &= \left[\sum_i \sum_j \beta_i^{(1)} \beta_j^{(1)} K(x_i, x_j) + \sum_i \sum_j \beta_i^{(2)} \beta_j^{(2)} K(x_i, x_j) \right. \\ &\quad \left. 2 \sum_i \sum_j \beta_i^{(2)} \beta_j^{(2)} K(x_i, x_j)\right]^{\frac{1}{2}} \qquad (6.11)\end{aligned}$$

The range of the projections $R_{X,d}$ of a set of examples X in given directions $d = x^{(1)} \quad x^{(2)}$, $x^{(j)} = \sum_i x_i \beta_i^{(j)}$ for $j = 1, 2$ would be

$$R_{X,d} = [\min_{x_i \in X} P_{x_i,\, d}, \max_{x_i \in X} P_{x_i,\, d}]. \tag{6.12}$$

Given a set $X = \{x_1, \ \cdots, x_n\}$, the skin segment with an angle θ around $d = x^{(1)} \quad x^{(2)}$ is

$$SS_{X,d} = \left\{ x_i \ | x_i \in E(X), \ \cos^{\ 1} \frac{\langle d, x^{(2)} \quad x_j \rangle}{\|d\| \cdot \left\| x_i^{(2)} \quad x_j \right\|} \leq \theta \right\}. \tag{6.13}$$

The angle θ_j between vectors $x_j \in X_i$ and center of gravity $x_i^{(g)}$ and $w = x_+^* \quad x^*$ can be found using

$$\left\langle w, x_i^{(g)} \ {}^{x_j} \right\rangle = \|w\| \cdot \left| \left| x_i^{(g)} \quad x_j \right| \right| \cos\theta, \text{ and } \theta = \cos^{\ 1} \frac{\langle w, x_i^{(g)} \quad x_j \rangle}{||w|| \cdot \ \left| \left| x_i^{(g)} \quad x_j \right| \right|}. \tag{6.14}$$

The decision boundary is perpendicular to $w_2^* \quad w_1^*$, where w_1^* and w_2^* are the nearest points between the RCHs. Gilbert's algorithms[15] are used to identify the nearest points between convex hulls (as shown in Algorithm 3). Figure 6.2 illustrates an example of GISVM for a linearly nonseparable case. The two classes are enclosed with convex hulls and overlap, as shown in Fig. 6.2(a). The skin of the convex hull is shown in Fig. 6.2(b). Two RCHs are highlighted with solid triangles: the outer RCHs transform the problem into a linearly separable case [see Fig. 6.2(c) for an example], whereas the inner RCHs define the skin thickness. The skin for retention is depicted in Fig. 6.2(d).

Algorithm 3: Gilbert's algorithm for finding the nearest points of two convex hulls.[15]

```
1. $Z \leftarrow \{\phi_+ \quad \phi \ |\phi_+ \in \Phi_+, \phi \ \in \Phi \ \}$
2. Randomly select $z^* \in C(Z)$
3. Repeat
4. $z_{\text{old}}^* \leftarrow z^*$
5. $z \leftarrow \arg\min_{z_i \in Z} P(z_i, z*)$
6. $z \leftarrow \arg\min_{z_i \in Z} P(z_i, z*)$
7. Until $\left|\left| z^* \quad z_{\text{old}}^* \right|\right| \approx 0$
```

6.4 Experimental Results and Discussion

6.4.1 Synthetic and benchmark data preparation

The experiments presented here used synthetic data sets, real-world data sets, and CE videos for evaluation. Two synthetic data sets were created by randomly sampling 2D Gaussian functions and the checkerboard function, namely the XOR data set (see Fig. 6.3 for examples). Ten sets of examples

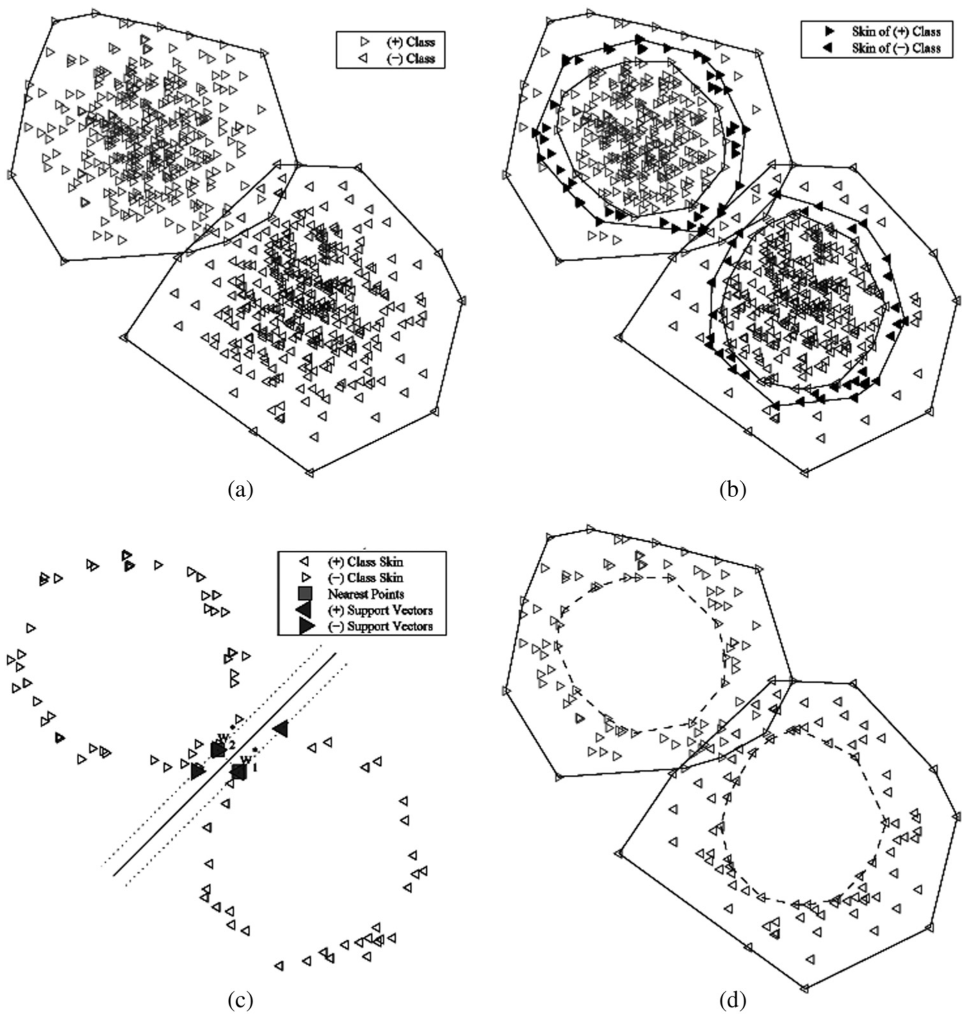

Figure 6.2 Geometric incremental SVM using convex hull skin for linearly nonseparable problems. (a) Two linearly nonseparable classes consisting of 700 examples. (b) The classes become linearly separable using $RCH(\mu = 0.1)$. (c) The decision boundary is found by finding the nearest points between two RCHs. (d) Examples within $S(x_i, 0.1, 1)$ are retained for future model updates.

were randomly generated using each model. The Gaussian data set has 1% overlap, whereas the XOR data set has no overlap. Four real-world data sets were obtained from the UCI machine learning repository.[1] In addition, a mammogram[8] data set was used. Each feature in a data set was normalized to unify its range to between 0 and 1. Table 6.2 lists the properties of the benchmark data sets used in the experiments.

6.4.2 Parameter selection

Figure 6.3 illustrates the decision boundaries of the proposed method applied to synthetic data sets using RBF, polynomial, and linear kernels. The shade in

Table 6.2 Benchmark data sets and their characteristics.

Data Sets	Number of Dimensions	Positive Class	Data Set Size	
			+ Class	Class
SPECT	22	1	212	55
PIMA	8	1	268	500
YEAST	8	CYT	463	1021
IONOSPHERE	34	b	126	225
MAMMOGRAM	6	2	260	10923

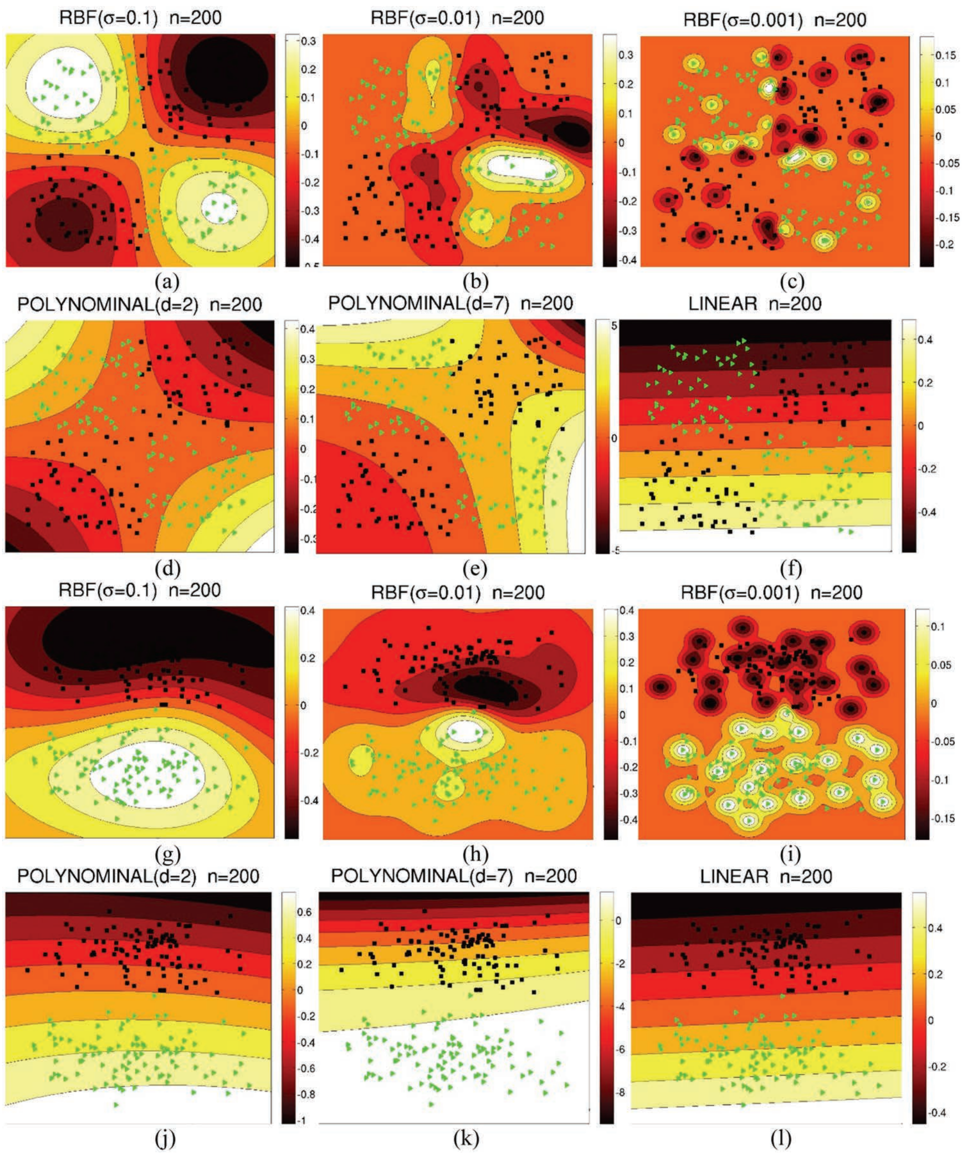

Figure 6.3 Decision boundaries from applying the proposed method to the synthetic data sets using different kernels.

the plots depicts the distance to the decision boundary. The incremental training starts with ten examples, and with each update, five new examples are randomly selected and used. The updates continue until all examples are exhausted. As shown in the figure, when the σ of the RBF kernel is decreased, the final classifier appears overfitted. Among all of the kernels tested, RBF kernels with $\sigma = 0.1$ resulted in better decision boundaries. It is evident that when examples are presented to the GISVM in an incremental fashion, the proposed method achieves superior closeness in modeling the underlying data distribution. The optimal level is reached with RBF kernels of $\sigma = 0.1$.

6.4.3 Efficiency analysis

Figures 6.4(a) and (c) illustrate the maximum number of examples retained (i.e., the examples in the skin of the RCHs) for various σ used in RBF kernels. As σ decreases, the number of examples retained significantly increases. When σ reaches 0.01, the number of examples retained is approximately the total number of examples in the training set. This indicates overfitting of the model, which is consistent with previous experimental results. On the other hand, the number of support vectors (in solid curve) varies slightly. According to these plots, a good choice for σ is 0.1, which provides a close description of the data sets and retains only a small number of examples in the training iterations.

Figures 6.4(b) and (d) show the number of examples retained in the training iterations. With a properly chosen σ, the number of retained examples is small, which implies the stability of the incremental learning

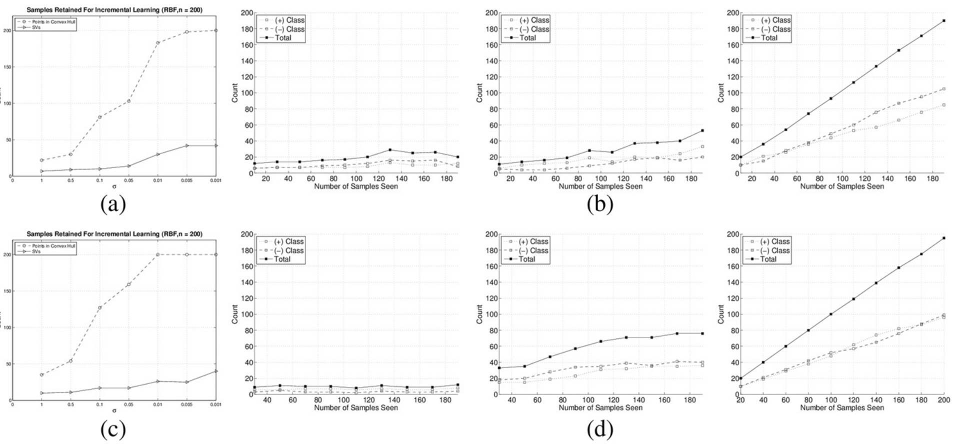

Figure 6.4 The number of extreme points identified using RBF kernels. (a) and (c) show the number of retained examples and SVs as a function of σ using Gaussian and XOR data sets, respectively. (b) and (d) show the number of retained examples using RBF kernel when $\sigma = 1.0, 0.1,$ and 0.01 (from left to right).

process. This also indicates that a much-smaller amount of memory was used to complete the learning.

Table 6.3 lists the time (in seconds) required by the GISVM and batch-learning libSVM to complete the training. Ten repetitions were conducted, and the average time and the standard deviation are reported. Because random examples were used, the training time varies. These experiments assumed that an equal number of examples was used to update a classifier in the incremental learning. The size of examples is referred to as step size Δ.

Limited by the number of examples in the benchmark data sets, two step sizes (i.e., $\Delta = 10$ and $\Delta = 20$) were used in the evaluations. It is clear that the time used by incremental learning is much less than that used by the libSVM. Among all cases, the MAMMOGRAM case consists of the largest number of examples and took significantly more time for training. Although the minimum time to complete training using MAMMOGRAM is in the time range of the GISVM, it can require up to triple the time that the libSVM needs. It is evident that the proposed incremental learning handles data efficiently and can update the classifier in much less time. The average time cost for this method to complete is approximately 13.4% of the time cost for batch-learning SVMs.

It is an interesting observation that a larger step size does not necessarily result in a longer training time. For data sets SPECT, PIMA, and MAMMOGRAM, training of the GISVM took less time using a step size of 20 than a step size of 10. Even in the other cases, the difference is small. This is probably due to the fact that only a small number of examples (i.e., examples within the skin of the RCHs) were carried over to the next round of updates.

Table 6.3 The average training time (in seconds) and standard deviation using batch-learning libSVM and the proposed GISVM. The number of iterations is also reported for the GISVM.

Data Sets	**Batch-Learning libSVM**	**GISVM**			
		Δ 10		**Δ 20**	
	Time	**Time**	**Iteration**	**Time**	**Iteration**
GAUSSIAN	2.9 (0.4)	0.2 (0.04)	19	0.21 (0.06)	9
XOR	5.2 (0.4)	0.34 (0.03)	19	0.54 (0.08)	9
SPECT	2.7 (0.3)	0.35 (0.05)	11	0.25 (0.02)	6
YEAST	97.2 (12.5)	3.59 (0.25)	73	5.16 (0.42)	37
PIMA	71.4 (7.9)	21.14 (0.25)	36	13.39 (1.34)	19
IONOSPHERE	3.8 (0.5)	0.3 (0.04)	15	0.43 (0.05)	8
MAMMOGRAM	4787 (2261)	2859 (541)	445	1540 (338)	223

6.4.4 Accuracy analysis

Figure 6.5 illustrates the classifiers' performance based on accuracy, sensitivity, and specificity during the incremental iterations. Ten repetitions were conducted with randomized initial examples. In each data set, 50% of the data were used for training; the remaining examples were used for testing.

In each case, a SVM classifier was created using all the training data. The best parameters were selected based on their generalization performance with

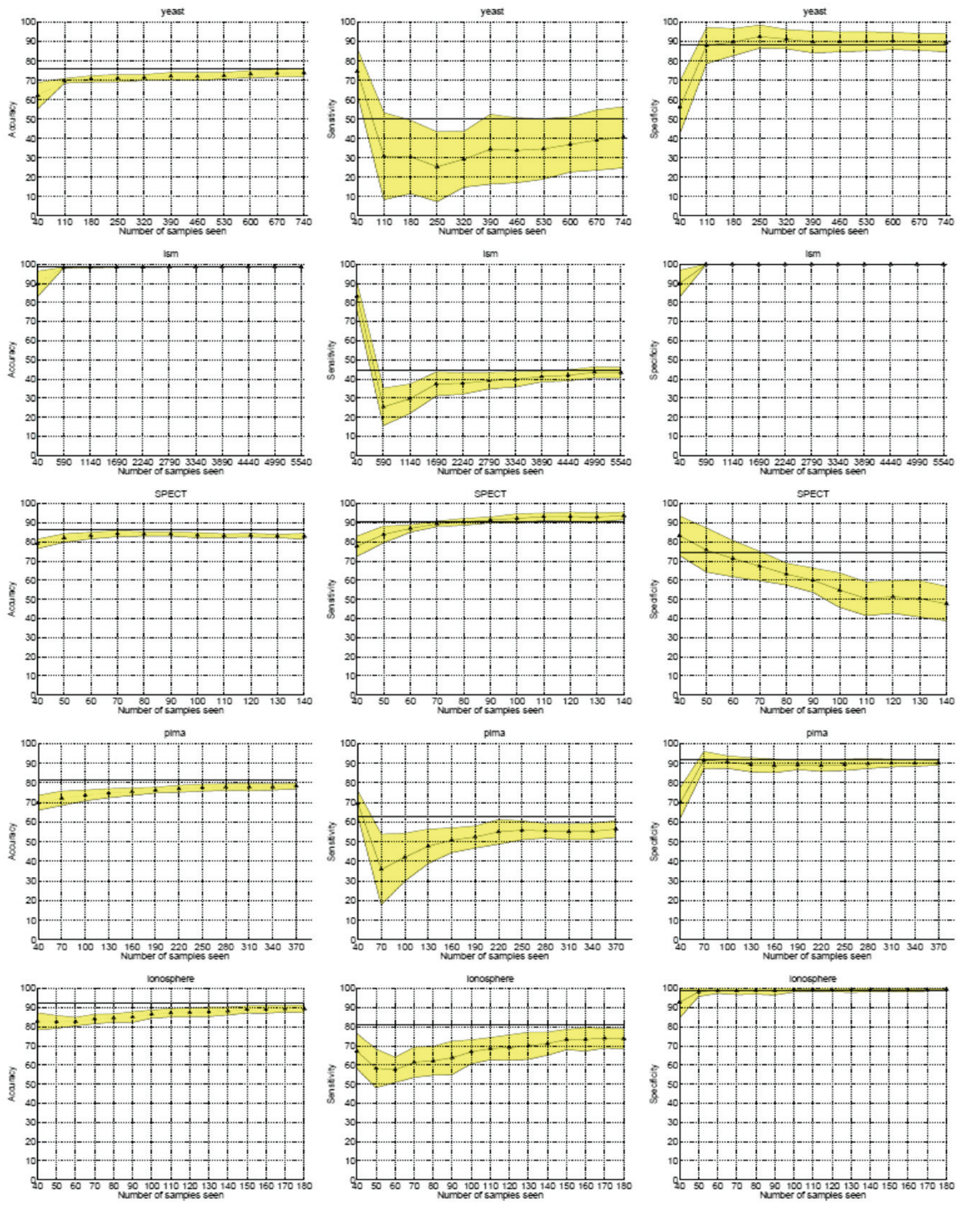

Figure 6.5 Accuracy performance of the GISVM using UCI data sets.

Table 6.4 Parameters used in the proposed GISVM.

Data Sets	Kernel Parameter		Convex Skin	
			μ_u	μ_l
SPECT	σ	0.05	0.9	0.6
YEAST	σ	0.10	0.9	0.6
PIMA	σ	0.15	0.5	0.3
IONOSPHERE	σ	0.01	0.5	0.3
MAMMOGRAM	σ	0.01	0.8	0.4

the testing data set. Table 6.4 lists the selected kernels and parameters that gave the best performance measures. The results from these classifiers are used as a reference and are depicted as the horizontal lines in Fig. 6.5.

In the proposed incremental learning process, ten examples were randomly selected from each class of the training set, and a SVM was trained. In each incremental step, ten randomly selected examples from the remaining training data set were used to update the classifier. The intermediate classifiers were evaluated with the test data set. For each data set, ten repetitions were conducted, and the average performance is plotted with a solid line in Fig. 6.5. The shaded area depicts the accuracy variation.

With more examples included in the training process, the classifier trained with the proposed method improves its performance; this is evident in the cases of YEAST, SPECT, PIMA, and IONOSPHERE. In the case of MAMMOGRAM (i.e., ISM), the performance is already close to optimal at the beginning, and there is no room for improvement. However, improvement in sensitivity can still be observed in the training, and by the end of iterations, the classifier outperformed the batch learning by a small margin.

Despite a slight drop of specificity of the SPECT data set, the SVMs trained with the proposed method achieved the same performance or even outperformed the batch-learning method. As listed in Table 6.2, the SPECT data set contains more positive examples than negative ones, the ratio of which is approximately 4:1. Hence, the improvement of sensitivity leverages the underperformance of specificity, and the overall accuracy is close to the batch-learning results. It is interesting that in five cases, the intermediate classifier had degradation in early iterations, but the training process was able to recover to the benchmark performance asymptotically as additional examples are included.

6.4.5 Experiments with CE videos

The analysis tool provided by the manufacturer of the Pillcam® capsule endoscope plots the path of the device through the digestive tract based on the wireless signal strength transmitted to the external image downloader carried by the patient. Our experiments on CE videos were performed to automate the classification of the frames in CE videos into digestive organs, namely the esophagus, stomach, small intestine, and colon.

Six CE videos were collected and manually annotated by gastroenterologists. Each video consists of approximately 55,000 frames. Out of the six videos, one was randomly selected to train the classifier, and the other five were used for testing.

In previous experiments with CE videos, the HSV color space was found to have a better classification performance on average.[16] In addition, using the histogram significantly reduces the dimensionality.* Hence, the color histogram in the HSV space was adopted as a feature. The color histogram is a very large and sparse matrix: With n bins used in each color component, there are n^3 features using the HSV histogram for every video frame, most of which are zeros or close to zeros. To suppress sparseness and the number of values in features, only the hue and saturation (HS) components were used. As observed in previous experiments,[16,29] using HS components improves control of lighting variations in the GI tract.

The order of classification of multiclass SVMs was determined based on the preliminary evaluation. In the experiments, identification of the esophagus gives the best accuracy followed by the identification of the small intestine. Hence, the order is determined and listed in Table 6.5. The kernels used to train a SVM are also included in this table.

In the learning process, 50 frames were randomly selected from each class of the training video to train a SVM. In each incremental step, 20 frames randomly selected from the remaining training video frames were used to update the classifier. The iteration repeats until the training examples exhaust. Table 6.6 lists the accuracy of the final classifiers. The performance of this method is highly satisfactory. With the majority of frames acquired in the stomach and small intestine, the average accuracies are 86.9% and 94.4%, respectively. Images acquired in the colon are disturbed by the presence of feces.

At the end of incremental training, only 12% of the frames were part of the skins among the four classes for the hierarchical SVMs. Apparently, the

Table 6.5 Order and parameters of hierarchical classification of organs of CE videos.

Order	Dividing Classes	Kernel Parameters
1	Esophagus vs. the rest	RBF (σ 0.15)
2	Small intestine vs. stomach and colon	RBF (σ 0.1)
3	Stomach vs. colon	RBF (σ 0.5)

* Each frame is a 256×256 color image. If color is used, the dimensionality of each example is up to 196,608.

Table 6.6 GISVM performance of digestive organs in CE videos.

Video	Esophagus	Stomach	Small Intestine	Colon
1	100%	87.6%	94.2%	85.3%
2	94.4%	85.8%	95.3%	82.2%
3	95.0%	87.2%	94.7%	84.3%
4	100%	86.4%	94.1%	83.7%
5	90.0%	87.7%	93.9%	94.3%

smaller number of examples demands much less memory space for the learning process and thus provides a plausible mechanism for handling a large amount of data. When new examples are added, the classifier is updated efficiently in contrast to the conventional batch-learning methods.

6.5 Conclusion

This chapter presents a GISVM method to learn from large data sets with an emerging trend and dynamic patterns. To overcome high computational demands from a large data set, this method identifies a subset of examples for the training process. It extends the reduced convex hull concept and defines the skin segments of convex hulls. The skin is found by identifying the extreme points of the convex hull. This method is founded on the idea that the examples within the convex hull skin are a superset to the support vectors, including the potential ones in future training. When additional examples are provided, they are used together with the skin of the convex hull constructed from the previous data set.

Using the skin of convex hull in the incremental learning process results in a small number of instances at every incremental step. The set of extreme points are found by recursively searching along the direction defined by a pair of extreme points. Besides the advantages in computational efficiency, the proposed method handles linearly nonseparable cases in multiclass problems.

Experiments were conducted with synthetic, benchmark, and CE data sets. With the synthetic data sets, the proposed method achieves highly satisfactory classifiers that closely model the underlying data distribution with appropriate kernels. The choice of RBF kernel for the synthetic data sets provides a good description of the data sets and retains only a small number of examples in the training iterations.

Using the experiments on benchmark data sets, this chapter demonstrates that the GISVM learning handles data efficiently and updates the classifier in approximately 13.4% of the time needed by the batch-learning SVM. Performance over the incremental steps further verifies the superior stability, improvement, and recoverability of the proposed method. The accuracy over the incremental steps increases steadily. Even in the cases when the performance drops, the classifier is able to recover to previous levels in future iterations because the convex hull skin that contains useful examples is retained. Furthermore, the improvement in the performance measures over the incremental steps (by deleting examples other than the ones on the skin) indicates that retaining the examples on the skin preserves adequate information about the decision boundary of SVMs.

From the experiments on CE videos it was noted that the average performance of classifying a CE video is above 86.9%, which is very competitive. The amount of memory space required in the training process could be one-eighth of what is required by the conventional SVM, which casts new light on processing large data sets with limited resources. Further experiments on CE videos demonstrated that the GISVM can handle data that could not be handled by the libSVM.

References

1. A. Asuncion and D. J. Newman, University of California, Irvine, School of Information and Computer Sciences, UCI Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html (2007).
2. C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Software* **22**(4), 469–483 (1996).
3. D. J. C. Barbosa, J. Ramos, and C. S. Lima, "Detection of small bowel tumors in capsule endoscopy frames using texture analysis based on the discrete wavelet transform," *Proc. IEEE Eng. Med. Biol. Soc.*, 3012–3015 (2008).
4. K. P. Bennett and C. Campbell, "Support vector machines: hype or hallelujah?" *SIGKDD Explor. Newsletter* **2**(2), 1–13 (2000).
5. H. A. Boubacar, S. Lecoeuche, and S. Maouche, "SAKM: Self-adaptive kernel machine; a kernel-based algorithm for online clustering," *Neural Networks* **21**(9), 1287–1301 (2008).
6. G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Proc. Advances in Neural Information Processing Systems*, 409–415 (2000).
7. C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," ACM Transactions on Intelligent Systems and Technology **2**(3), 1–27 (2011).
8. N. V. Chawla, K. W. Bowyer, L. O. Hall, and P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artificial Intelligence and Research* **16**, 321–357 (2002).
9. S. Cheng and F. Y. Shih, "An improved incremental training algorithm for support vector machines using active query," *Pattern Recognition* **40**, 964–971 (2007).
10. M. T. Coimbra and J. P. S. Cunha, "Mpeg-7 visual descriptors contributions for automated feature extraction in capsule endoscopy," *IEEE Trans. Circuits and Systems for Video Technology* **16**(5), 628–637 (2006).

11. D. J. Crisp and C. J. C. Burges, "A geometric interpretation of -SVM classifiers," NIPS: Advances in Neural Information Processing Systems, 12 (2000).
12. C. P. Diehl and G. Cauwenberghs, "SVM incremental learning, adaptation and optimization," *Proc. International Joint Conference on Neural Networks* **4**, 2685–2690 (2003).
13. C. Domeniconi and D. Gunopulos, "Incremental support vector machine construction," *Proc. IEEE International Conference on Data Mining*, 589–592 (2001).
14. E. Parrado-Hernandez, I. Mora-Jimnez, J. Arenas-Garca, A. R. Figueiras-Vidal, and A. Navia-Vzquez, "Growing support vector classifiers with controlled complexity," *Pattern Recognition* **36**(7), 1479–1488 (2003).
15. E. G. Gilbert, "An iterative procedure for computing the minimum of a quadratic form on a convex set," *SIAM J. Control* **4**(1), (1966).
16. B. Giritharan, X. Yuan, J. Liu, J. Oh, and S. Tang, "Bleeding detection from capsule endoscopy videos," *Proc. IEEE Eng. Med. Biol. Soc.*, 4780–4783 (2008).
17. R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Process. Lett.* **2**(1), 18–21 (1973).
18. Y. S. Jung, Y. H. Kim, D. H. Lee, and J. H. Kim, "Active blood detection in a high-resolution capsule endoscopy using color spectrum transformation," *International Conference on Biomedical Engineering and Informatics*, 859–862 (2008).
19. A. Karargyris and N. Bourbakis, "Identification of ulcers in wireless capsule endoscopy videos," *Proc. 6th IEEE International Conference on Symposium on Biomedical Imaging*, Piscataway, NJ (2009).
20. A. Karargyris and N. Bourbakis, "Identification of polyps in wireless capsule endoscopy videos using log gabor filters," *Life Science Systems and Applications Workshop*, 143–147 (2009).
21. S. Katagiri and S. Abe, "Incremental training of support vector machines using hyperspheres," *Pattern Recognition Lett*. **27**(13), 1495–1507 (2006).
22. S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "A fast iterative nearest point algorithm for support vector machine classifier design," *IEEE Trans. Neural Networks* **11**(1), 124–136 (2000).
23. R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," *Proc. 17th International Conference on Machine Learning*, 487–494 (2000).
24. V. Kodogiannis, "Computer-aided diagnosis in clinical endoscopy using neuro-fuzzy systems," *IEEE International Conference on Fuzzy Systems* **3**, 1425–1429 (2004).

25. V. S. Kodogiannis and M. Boulougoura, "An adaptive neurofuzzy approach for the diagnosis in wireless capsule endoscopy imaging," *International J. Information Technology* **13**(1), 46–54 (2007).
26. P. Y. Lau and P. L. Correia, "Detection of bleeding patterns in WCE video using multiple features," *Proc. IEEE Eng. Med. Biol. Soc.*, 5601–5604 (2007).
27. B. Li and M. Q.-H. Meng, "Analysis of wireless capsule endoscopy images using chromaticity moments," *IEEE International Conference on Robotics and Biomimetics*, 87–92 (2007).
28. B. Li and M. Q.-H. Meng, "Computer-aided detection of bleeding regions for capsule endoscopy images," *IEEE Trans. Biomed. Eng.* **56**(4), 1032–1039 (2009).
29. J. Liu and X. Yuan, "Obscure bleeding detection in endoscopy images using support vector machines," *Optimization and Engineering* **10**(2), 289–299 (2009).
30. M. E. Mavroforakis and S. Theodoridis, "A geometric approach to support vector machine classification," *IEEE Trans. Neural Networks* **17**(3), 671–682 (2006).
31. M. E. Mavroforakis, M. Sdralis, and S. Theodoridis, "A geometric nearest point algorithm for the efficient solution of the SVM classification task," *IEEE Trans. Neural Networks* **18**(5), 1545–1549 (2007).
32. P. Mitra, C. A. Murthy, and S. K. Pal, "Data condensation in large databases by incremental learning with support vector machines," *Proc. 15th International Conference on Pattern Recognition* **2**, 708–711 (2000).
33. P. Mitra, C. A. Murthy, and S. K. Pal, "A probabilistic active support vector learning algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence* **26**(3), 413–418 (2004).
34. F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, "On-line independent support vector machines," *Pattern Recognition* **43**(4), 1402–1412 (2010).
35. X. Peng and Y. Wang, "CCH-based geometric algorithms for SVM and applications," *Appl. Mathematics and Mechanics* **30**(1), 89–100 (2009).
36. C. Renjifo, D. Barsic, C. Carmen, K. Norman, and G. S. Peacock, "Improving radial basis function kernel classification through incremental learning and automatic parameter selection," *Neurocomput.* **72**(1–3), 3–14 (2008).
37. A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi, "Incremental training of support vector machines," *IEEE Trans. Neural Networks* **16**, 114–131 (2005).
38. N. A. Syed, H. Liu, and K. Kay, "Incremental learning with support vector machines," *Proc. Workshop on Support Vector Machines at the International Joint Conference on Arterial Intelligence* (1999).

39. F. Vilarino, L. Kuncheva, and P. Radeva, "ROC curves and video analysis optimization in intestinal capsule endoscopy," *Pattern Recognition Lett.* **27**, 875–881 (2006).

40. S. Agarwal, V. V. Saradhi, and H. Karnick, "Kernel-based online machine learning and support vector reduction," *Neurocomput*. 71, 1230–1237 (2008).

Xiaohui Yuan received his B.S. degree in Electrical Engineering from Hefei University of Technology, China in 1996 and his Ph.D. degree in Computer Science from Tulane University in 2004. After graduating, he worked at the National Institutes of Health as a postdoctoral research scientist from 2005 to 2006 and then joined the University of North Texas in 2006, where he is currently serving as an Associate Professor in the Department of Computer Science and Engineering. He is a recipient of the Ralph E. Powe Junior Faculty Enhancement Award in 2008 and the Air Force Summer Faculty Fellowship in 2011 and 2012. He is a member of IEEE and SPIE. His research interests include computer vision, pattern recognition, data mining, and artificial intelligence.

Balathasan Giritharan received his B.S. degree in Computer Science from the University of Peradeniya, Sri Lanka in 2002, and his M.S. in Computer Science (2007) and Ph.D (2012) in Computer Science and Engineering from the University of North Texas. From 2002 to 2005, he worked as a lecturer at the University of Peradeniya. He interned at Abbott Laboratories in 2011 and then joined CityGrid Media as a Data Scientist in 2012. His research interests include machine learning, image processing, and business intelligence.

Mohamed Abouelenien received his B.S. and M.S. degrees in Electronics and Communication Engineering in 2005 and 2008 from the Arab Academy for Science and Technology in Alexandria, Egypt. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at the University of North Texas, where he has been appointed as a teaching and research assistant since 2009. His research interests include machine learning, ensemble classification, image processing, pattern recognition, facial recognition, computer vision, and dimensionality reduction.

Jianguo Liu received his B.S. and M.S. degrees in Computational Mathematics from Xiangtan University, China, and his Ph.D. in Applied Mathematics from Cornell University. He currently teaches in the Department of Mathematics at the University of North Texas. His research interests include optimization, numerical analysis, and machine learning.

Xiaojing Yuan received her B.S. degree in Electrical Engineering from Hefei University of Technology, China in 1994, her M.S. degree in Computer Engineering from the University of Technology and Science of China in 1997, and her Ph.D. in Mechanical Engineering from Tulane University in 2003. After working as a postdoctoral researcher with the Image Understanding Lab at Tulane, she joined the University of Houston as a Visiting Assistant Professor in 2004. She is an Associate Professor in the Engineering Technology Department at the University of Houston. Dr. Yuan is a senior member of IEEE and a member of ASEE and ISA. She has over 70 publications in peer-reviewed journals, book chapters, and conferences. Her research and teaching interests include pattern recognition, computational intelligence, image processing, data mining, and information processing in wireless sensor networks.