Abstract

Sun, Xuejun

ULEDS-SVMs: Upper/Lower Limit and Error Data Supported Support Vector Machines

(Under the direction of Jon Doyle)

A Support Vector Machine, ULEDS-SVM, was developed for classification in data domains which contain limits or errors.

Data with upper or lower limits are not the same as missing data. They may provide constraints at a certain level in data classification and modeling. Data with errors may be regarded as the special co-existing case of an upper and a lower limit at the two boundaries of an attribute. Data with such qualities widely exist, from scientific data measurements to databases resulted from integration. Including these data in training rather than dropping them or replacing them with some arbitrary value is desirable because they may provide useful constraints.

An enhanced 1R algorithm is described which may be able to handle a data domain that contains limits and errors. However, the time complexity is $O(mk^n)$, where m is the number of entities, n the number of attributes, and k the number of elements in one attribute. This does not allow good scalability.

Support Vector Machine (SVM) treatment of the data in such a domain is, however, very promising. The mathematical foundation for treating this kind of problem is provided herein by recognizing the feasibility concepts for training and prediction in SVMs. It was found that

an entity of data items is not trainable if the class, y, and the normal of the separation plane, (w, γ), satisfy some conditions: an attribute μ exists such that $w_\mu * y < 0$; $w_\mu * y > 0$; $({}_\mu\hat{x}' * w - \gamma)({}_\mu\check{x}' * w - \gamma) < 0$ for an upper limit, lower limit and an error, respectively, where ${}_\mu\hat{x}$ and ${}_\mu\check{x}$ are data at their centroid values, except for the μth component which is located at the upper and lower boundaries of an error, respectively. Untrainable data need to be dropped, and trainable data need to be included in training.

By utilizing an intersection concept, which is the intersection of the separation plane and the uncertain region of limit or error data, testing can be set to be correct if such an intersection exists for limit or error data. Intersection conditions are determined by the existence of an attribute, μ, for an entity such that $f(x) * w_\mu * U_\mu > 0$ for limits, or $f({}_\mu\hat{x}) * f({}_\mu\check{x}) < 0$ for errors, where $f(x) = x * w - \gamma$ is the function of the separation plane, and U_μ denotes the upper (1) or lower (-1) limit at the μth component. The prediction of class is meaningful for those data without intersections.

An astronomical data set (CHDF-N) is integrated based on Chandra Deep Field (CDF) and Hubble Deep Field (HDF) North observations. Some spectroscopic identifications are available, but some are beyond the existing instrument limits. CHDF-N is used as a test bed for the ULEDS-SVMs algorithms application implemented via Matlab. This classification may be applicable to the study of the formation and evolution of galaxies in the deepest universe.

The separation of stars and extragalactic objects allows a 100% accuracy rate; however, the separation would otherwise be ambiguous if the limit data in the extragalactic class were not included. Training and testing using the leave-one-out partition method achieved an 82% ac-

curacy for the separation of galaxies and active galactic nuclei (AGNs). This rate is better than the 72.4% accuracy rate attained using the conventional $R - log(F_x)$ plot separation method commonly used in the astronomy community. The prediction rate increased from 49.6% using conventional SVMs to 75.5% using ULEDS-SVMs.

ULEDS-SVMs: Upper/Lower Limit and Error Data Supported Support Vector Machines

by

XUEJUN SUN

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of

Master of Science

COMPUTER SCIENCE

Raleigh, North Carolina

2003

APPROVED BY

_____ _____

_____

Chair of Advisory Committee

Biography

Name: Xuejun Sun

Education:

2002-2003 M.S., Computer Science, NC State University

2001-2002 Student for life long education program, Computer Science, NC State University

1984-1987 M.S., Physics, Graduate School, University of Science and Technology of China

1980-1984 B.S., Physics, Beijing Normal University

Research and visiting experiences:

1999-2001 Staff, Physics, University of Alabama in Huntsville

1998-1999 Visiting Scholar, Universities of Space Research Association/NASA Marshall Space Flight Center

1997-1998 Visiting Scholar, Los Alamos National Lab

1996-1997 Staff, Beijing Astronomical Observatory, Chinese Acadamy of Sciences

1987-1996 Staff, Institute of High Energy Physics, Chinese Acadamy of Sciences

1994-1996 Visiting Scholar, Max-Plank-Institut für extraterrestresche Physik, Germany

1990-1991 Visiting Scholar, Institute of Physical and Chemical Research, Japan

ACKNOWLEDGEMENTS

The author is very grateful for the support and help of so many people during the time that he conducted his research work and thesis preparation.

The author's advisor, Dr. Jon Doyle, guided the Kernel Method direction for the author's thesis. His conversations were always inspirational and his encouragement accompanied the author's work throughout this research.

Dr. Shuangnan Zhang's discussions with the author about the current status of major surveys in astronomy helped the author to make the decision to choose Chandra and Hubble Deep Field as the area for the Machine Learning application that uses the method developed in this research. The confidence to work on this application was also encouraged by Dr. Dick Shaw.

The author's interest in Machine Learning largely originated from communications with Dr. Fenimore. His support and inspiration helped the author significantly, not only during the author's student life but beyond. Dr. Xiaohui Fan's discussions with the author about the Slon Digital Sky Survey strengthened the author's ideas about the usefulness of a Machine Learning method that features an upper limit and error handling capability.

Mr. Lloyd Williams and Ms. Mary K.S. Brown edited the author's English. Mr. Brent Dennis helped the author in LaTex.

The author's wife, Toni, and daughter, Anna, encouraged the author throughout his graduate study at North Carolina State University. Without their understanding and steadfast support he could not have finished the course work, research and thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In real world environments that include data measurements in scientific research, a result may be obtained with an upper limit value rather than a definite one. Even though a value is obtained, it may contain significant errors. It becomes difficult or even impossible to improve the quality of the data further because the result obtained has either reached the current technological limit in determining the accuracy of an attribute, or it has already been archived, or improving it would be too costly. Technically, every numerical measurement contains some errors. The problem to be addressed, therefore, is to determine if one can neglect the error. More often than not, however, including data with upper and lower limits and errors may provide more meaningful information or imply some constraints that would not be evident if the errors were dropped. In Fig. 1.1, a simple case is used to demonstrate the usefulness of including these kinds of constraints.

Suppose there are two points at A(-1/2, 1/2) and B(1/2, -1/2) with classes of -1 and +1,

respectively. For a binary class problem, +1 and -1 are convenient designations for their class not only because of the sign discrimination, but also their fit into a formula for mathematical treatment, as discussed in Chapter 8. Hereafter, +1 and -1 will be used as the binary classes unless specifically stated otherwise. To classify them correctly, a separation plane is drawn as the dash-dot line in Fig. 1.1 which satisfies their classifications. However, if there is another data item which has an upper boundary on the vertical axis at C(3/4, 1/2) with class +1, then it makes a difference whether one drops or includes it in determining the separation plane where the vertical line below point C indicates that it is an upper limit with a boundary of 1/2 on its vertical axis. In fact, the solid horizontal line separates the classes better than the dash-dot line which misclassifies data C.

Data integration, or merge, may encounter different precision issues when more than one of the primary databases do not match up properly in terms of accuracy. This phenomenon also raises questions about precision. This integration trend can be expected to increase along with the increase of accessing and processing databases via the Internet. Treatment of nonuniform precision or hybrid data, therefore, needs to be addressed.

Many Machine Learning (ML) methods, from the simplest 1R rule [12] to more sophisticated ones such as Support Vector Machines (SVMs [25]), recognize only data presumed to be without any error or upper and lower limits. This limitation restricts the application of these methods. Support Vector Machines (SVMs, [25]), representing one of the major kernel methods, are also powerful tools for data classification. A tutorial about SVMs can be found in [5]. A basic description of SVMs and a review of recent progress is also attached (see Chapter

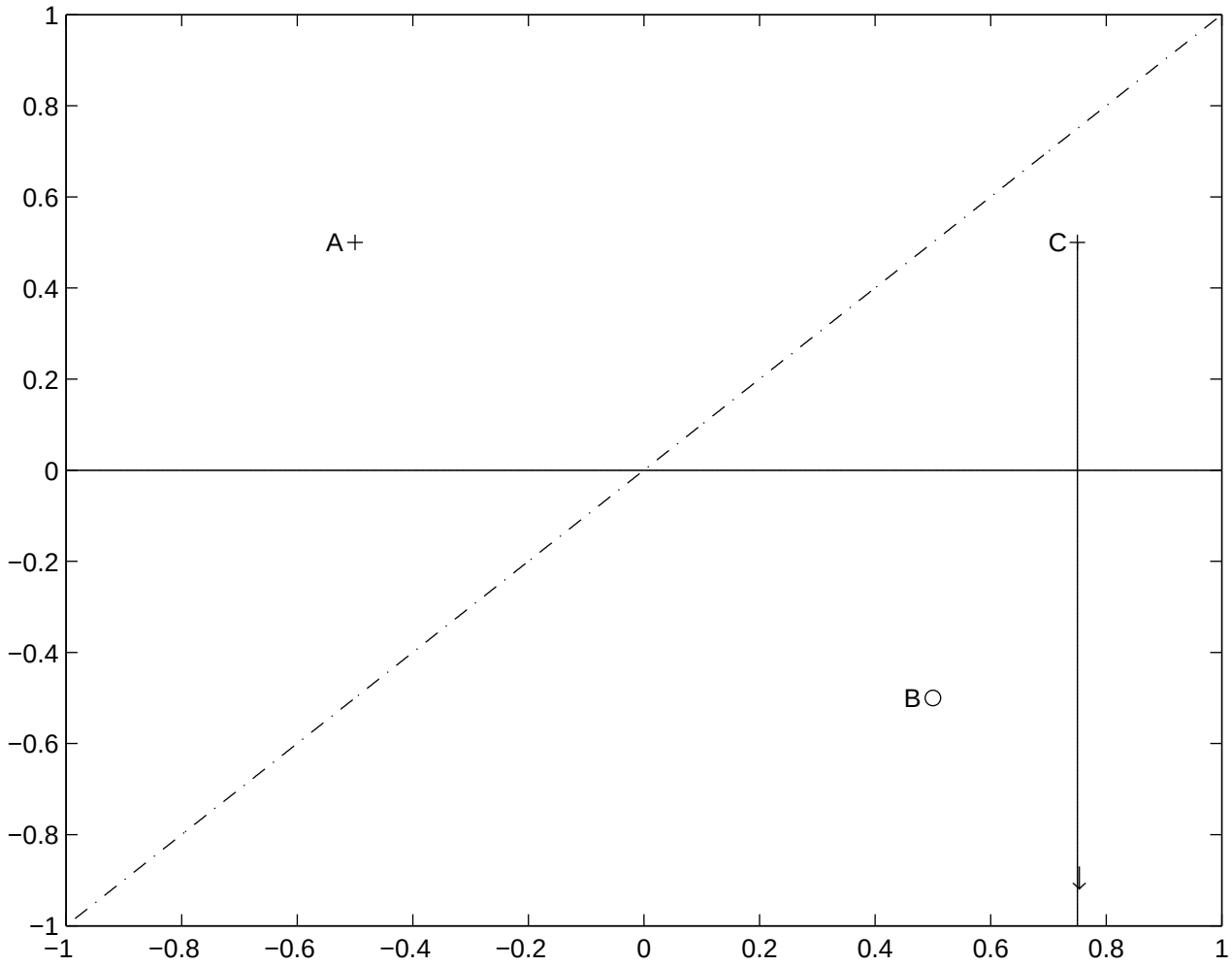

Figure 1.1: A simple two class classifi cation problem with upper limit data. Taking into account the information constrained by point C, whose data contain an upper limit on the Y axis, a classifi cation as the solid horizontal line would be appropriate. In contrast, dropping point C leads to the classifi cation, as indicated by the dash-dot line, that would misclassify point C by chance occurrence. ”+” and ”o” indicate the classes of +1 and -1.

8). This paper describes the linear SVMs developed in a domain that contains data with upper or lower limits and data with error. As a comparison, Chapter 2 describes a possible scenario with a 1R rule algorithm in such a domain. The treatment of this domain can be extended to many other statistical based ML regimes, such as the Naive Bayes Network, Decision Tree, etc. Treatments of such a domain are, however, quite different when using SVMs. Because of the unique feature of the SVM separation plane to discriminate classes, one can derive a concise mathematical formula to handle data conveniently in such a domain. A clean algorithm, based on this concise formula, is possible to treat data in such a domain. This paper focuses on SVM development in such domains in Chapters 3-6, which cover concepts, mathematical bases, algorithms and application in a data set. These SVMs are called ULEDS-SVMs (Upper/Lower Limit and Error Data Supported Support Vector Machines). The domains that contain data with upper or lower limits and errors are called ULED domains. The advantage of ULEDS-SVMs is obvious in terms of computing cost when comparing them to the enhanced 1R rule ML in a ULED domain. In Chapter 3 some theorems which will be used as the basis for treating data with upper or lower limits and data with errors in SVMs. Chapter 4 presents the generalized algorithms of ULEDS-SVMs.

There are several databases which are regarded as benchmarks in the AI community for algorithm testing and comparison, although none of these belongs to a ULED domain. Chapter 5 describes the integration of a data set from astronomical observations taken from Chandra Deep Field (CDF) and Hubble Deep Field (HDF) North, or CDHF-N for short. The classification of astronomical objects from these sky directions is an important issue in astronomy.

These observations are focused on the sky direction which has the least galactic and extragalactic obscuration and absorption. Long exposure times were devoted to image and spectroscopic observations in various energy bands, including radio, infrared, optical and X-ray sources, using the largest telescopes on the ground as well as in outer space in order to understand the distribution and evolution of galaxies. This field contains many sources identified by their spectral and morphological types. It also contains many unidentified sources whose further identifications are difficult, primarily because the instrument limits have been reached. In these sources, some attributes contain only upper limits, but statistical study is nonetheless worthwhile given the available data; other sources contain attributes which are relatively bright and saturated in image observation. So, only the lower limit in terms of brightness was obtained. A data set, CHDF-N, was generated from the sources collected from several literature sources and used as the test bed. A review of the literature and a description of the effort made regarding the data set integration are given in Chapter 5 and Appendix A. Chapter 6 and Appendix A describe the source classification results for training, testing and prediction using ULEDS-SVMs. Discussion and conclusions are provided in Chapter 7. A general review of SVMs may be found in Chapter 8. The code for the specific implementation of ULEDS Support Vector Machines is attached in Appendix B.

Chapter 2

Enhanced 1R - A Simple Method to Treat Data with Limit and Error Attributes

First, the definitions of upper and lower limits and of error are introduced. Such definitions help to eliminate ambiguity about the meaning of these terms and help to establish the mathematical foundation necessary to treat them.

Definition: Define $S \subseteq R^n$ as a data set over n variables or attributes. For a data point, $x \in S$, its μth attribute, x_μ, has an upper limit, $\hat{x}_\mu$, only if the μth attribute value, from its acquisition, is unknown but less than the upper limit, $x_\mu \leq \hat{x}_\mu$. $\hat{x}_\mu$ represents the upper limit along the μth component in R^n for the point.

An example of an upper limit of an attribute is found in the astronomical magnitude measurement of objects. Within a given period of exposure time and using a specific telescope mounted with a certain detection device, the magnitude of an object may be determined if it

is sufficiently bright. However, the object may be too faint to show up under such conditions. When it cannot be detected, one can nevertheless derive a magnitude as the object brightness limit, according to the observation specification and the background level of the direction. One cannot know how much fainter the object would be than the limit magnitude; however, one can say that the object is not brighter than the limit magnitude. Data with such an upper limit attribute frequently appear, especially in scientific data measurement.

Similarly one can define the lower limits:

Definition: for a point, $x \in S$, its μth attribute, x_μ, has a lower limit, $\check{x}_\mu$, only if the μth attribute, $x_\mu \geq \check{x}_\mu$. One refers to $\check{x}_\mu$ as the lower limit on the μth component in R^n for the point.

Sometimes there could be an error obtained for a variable or attribute.

Definition: for a point, $x \in S$, its μth attribute, x_μ, has an error, $\delta_\mu = \hat{x}_\mu - \check{x}_\mu$, only if there are both upper and lower limits on the μth component such that $\check{x}_\mu \leq x_\mu \leq \hat{x}_\mu$. When both equalities hold, $\delta_\mu = 0$, the data do not have any errors on the μth component. The value, x_μ, is called its centroid value of the μth component of point x, and $\check{x}_\mu$, $\hat{x}_\mu$ of x's lower and upper limit on the μth component, respectively. x is called the centroid point.

In the actual measurement of a quantity, the limit that is measured often refers to the limit of u at η confidence level or some σ level. Error is often expressed as $\hat{x}_\mu < x_\mu < \check{x}_\mu$ at η confidence level or some σ level. Quite often a contour around the error or limit can also be seen. This contour describes in terms of statistical uncertainty the limit value that is measured according to statistical distribution. Precise language is used to describe the fine structure of

the limits and errors. In most statistical learning processes, such a fine structure may not be required; a representative sharp boundary for limits is sufficient in many cases. The most frequent references to the confidence level of limits and errors are $1-\sigma$, $2-\sigma$, $3-\sigma$ or 90%. These are assumed to be true in the treatment of SVMs described in the following chapters; however, the fine structure of the errors and limits will be touched on a little later in this chapter for the enhanced 1R rule. More detailed statistical descriptions of errors and limits may be found in textbooks about statistics and data reduction.

Here it is worth noting that the upper or lower limit is not a missing datum. For missing data, one or more attributes are totally unknown. In contrast, the point with an upper or lower limit does tell that the value along a component does not exceed or go below a certain value, $\hat{x}_\mu$ or $\check{x}_\mu$. Including upper or lower limit data usually provides more information than a missing value would provide. Moreover, the constraint may be crucial to the support or elimination of some models.

2.1 1R Rule Extension

There is a very simple classification rule, called 1R, provided by [12]. A simple algorithm is implemented therein to treat data with limit and error attributes. This simple implementation has two purposes: the first is to put into practice a statistical learning approach by treating data with limits and errors; and second, it is to be used to compare the training approaches used here with the more sophisticated SVM methods. The 1R rule predicts that the classes will all be the same class as the most frequently occurring one [12]. This prediction technique appears to be

somewhat oversimplified. But, according to [12], the 1R rule is efficient and has reasonably good accuracy for most commonly used data sets. Details about its pseudocode can be found in Appendix A of [12].

The 1R rule may now be extended to cases that contain data with an upper limit. Given a point of data, (x, y), in the attribute space, $R^{n\times 1}$, $(x, y) \in R^{n\times 1}$, where y is the class of the item, assumes that the space is within the domain, $\{x|x_\nu^{min} \leq x_\nu \leq x_\nu^{max} \quad \forall\nu\}$, where the Greek subscript represents the attribute. If x has an upper limit along the μth attribute dimension, the probability of the x_μ distribution is $p(x_\mu)$, where

$$\int_{x_\mu^{min}\leq x_\mu\leq \hat{x}_\mu} p(x_\mu)dx_\mu = 1 \tag{2.1}$$

for the attribute with the upper limit of data (x, y). the probability for each element in the space $R^{n\times 1}$ can be obtained, and the 1R rule can be determined by using the largest probability in the space. For the case of the lower limit, the integration is over $\check{x}_\mu \leq x_\mu \leq x_\mu^{max}$.

In general, there is a probability, $p(x_{\mu 1}, x_{\mu 2}, ..., x_{\mu\ n})$, where

$$\int p(x_{\mu 1}, x_{\mu 2}, ..., x_{\mu\ n})dx_{\mu 1}dx_{\mu 2}...dx_{\mu\ n} = 1 \tag{2.2}$$

where $\mu 1, ..., \mu\ n$ are the attributes of the point, (x, y), with upper or lower limits.

The three-dimensional integer matrix, $COUNT(C, V, A)$ in [12], is the probability integration in space, $R^{n\times 1}$, at each corresponding element where C is the class and V is the value for the attribute, A. By extending the array into non-negative real values and integrating p

described above into COUNT, an upper and lower limit case can be cast.

Data attributes with errors can be seen as the existence of data bound by both upper and lower limits, $\check{x}_\mu \leq x_\mu \leq \hat{x}_\mu$. Integration is performed within these boundaries. Thus, all 1R rule algorithms of [12] can also be extended to data with error.

The probability distribution of the errors or the limits may not be known. Under such circumstances, an empirical distribution of a constant probability, $p = 1/(\hat{x}_\mu - x_\mu^{min})$, is assumed for one attribute upper limit case, $p = 1/(x_\mu^{max} - \check{x}_\mu)$ for a lower limit case, and $p = 1/(\hat{x}_\mu - \check{x}_\mu)$ for an error attribute.

The error or limit box is used to obtain the probability, $p = 1/VOB$, when there is more than one attribute involving errors or limits of a data point in space, $R^{n\times 1}$, where VOB represents the volume of the box of the error or limits occupied in the space, $R^{n\times 1}$. There is a probability of those data with limits or errors of $P(C,V,A) = VOE * \sum_{\mu_i:1-\mu_n,\forall item} p(x_\mu)$ to be in the element of class, C, with value, V, for the attribute, A, where VOE is the Volume of an Element in VOB in $R^{n\times 1}$.

The COUNTs of a class, C, having a value, V, for attribute, A, is $COUNT(C,V,A)$ for data without any errors or limits. The final probability of data that include limits or errors is $COUNT(C,V,A) = COUNT(C,V,A) + \lambda P(C,V,A)$, where $0 \leq \lambda \leq 1$ is a weighting parameter for limit and error. $\lambda = 1$ corresponds to the equal weighting of limit or error data with respect to data with definite values, while $\lambda = 0$ corresponds to the dropping of all data with errors or limits.

2.2 Comments

This simple treatment represents the current knowledge about data distribution in $R^{n\times 1}$ space, including both ordinary data with definite values and data with upper/lower limits and errors. Wherever there is a data entity with specific values of attributes, it falls into the specific element of the volume in $R^{n\times 1}$ space. Wherever there is an error or limit, a probability exists and is distributed somewhere in a volume box, VOB. Statistical learning based on this knowledge to treat data, including upper/lower limit and error data, is resumed. The most likely occurrence of the class throughout the training data set in 1R can thereby be obtained.

Alternatively, even without obtaining the most frequently occurring element class, the fractional frequency of occurrence in the event space may be extended to some other statistical learning methods based on probability distribution, such as the Naive Bayes network, Decision Tree, clustering analysis, etc.

Calculating COUNT for ordinary data entities is straightforward. However, calculating p and P and adding P to COUNT may be computationally expensive because it must be performed on each relevant element at a time. For a data point with limits or errors, the upper boundary of the time complexity caused by adding its P contribution to COUNT may extend over the whole domain in R. In cases where a large amount of data with limits and errors exists and where these boundaries are not well constrained, special treatment of the addition process is desirable in order to increase the speed. The acceleration algorithm may depend on the distribution of data within the database. Sparse data with limits and errors may use the algorithms directly. Databases containing large amounts of limit and error data may use the p integration

as a base, similar to the density probability technique. COUNT is then applied in addition to $P = \int pdx$. A drawback of such acceleration is the loss of the generality of the algorithms. The next chapter focuses on SVM treatment, which is not extendable from the strategy proposed in this section to treat limit and error data. However, the unique separation plane concept of SVM treatment provides a basis for a concise mathematical expression to discriminate classes although data with upper or lower limits and errors exist.

Chapter 3

Mathematical Foundation for ULEDS-SVMs

3.1 Training

According to conventional SVMs, a hyperplane divides two classes in a feature space so that misclassification is minimized while the margin is maximized (see, e.g. [5]). This phenomenon may be achieved, as shown in Figure 3.1, by optimizing the Lagrangian function,

$$\min_{(w,\gamma,\xi)\in R^{n+1+m}} \nu e'\xi + \frac{1}{2}w'w \ \text{ s.t. } D(Aw - e\gamma) + \xi \geq e,\ \xi \geq 0 \tag{3.1}$$

Here, w represents the normal to boundary planes of the support vectors, $x'w = \gamma \pm 1$, with a separation plane, $x'w = \gamma$; and γ is the scalar relevant to the distance of the plane to the coordinate system origin, ξ is the vector in R^m for the soft margin, D is the diagonal matrix of

the classes in $R^{m\times m}$, ν is the weighting parameter for the soft margin that serves to penalize the mix at the specified level, e is the ones vector, and the prime indicates the transpose of the corresponding vector or matrix. The dual to a linear SVM is

$$\min_{\alpha\in R^m} \frac{1}{2}\alpha' DAA'D\alpha - e'\alpha \text{ s.t. } e'D\alpha = 0, \quad 0 \le \alpha \le \nu e. \tag{3.2}$$

where α is the vector for Lagrangian multipliers (see, e.g., [5]). By default, a two-class problem has two classes dedicated as +1 and -1. The normal of the separation plane, w, is expected to point to the +1 class side. A data item is represented as $(x, y) \in R^{n\times 1}$, where x is represented by a vector for its attributes, and y is a scalar for its class.

To introduce the upper or lower limit or an error in a data attribute and to obtain a mathematical presentation of the treatment in such cases, the theorems below are introduced:

Theorem 1: An entity, (x, y), with upper (lower) limits is necessary for consideration in SVM training if and only if for every component, μ, among the components which contain upper (lower) limits for (x, y), the entity satisfies $w_\mu y > 0$ ($w_\mu y < 0$ for the lower limit), where w_μ is the μth component of the separation plane normal, and y is the sign of the class of the entity.

This theorem is one of the bases of the algorithms to be developed here. It is proved for the upper limit case only. Proof for the lower limit case is similar. According to $w_\mu y$, four configurations are possible, as shown in Table 3.1:

a) when w_μ is positive and y is also positive, the "+1" class is expected to be above the plane along the μth component. But the point, x, has an upper limit, $\hat{x}_\mu$. It would not satisfy the clas-

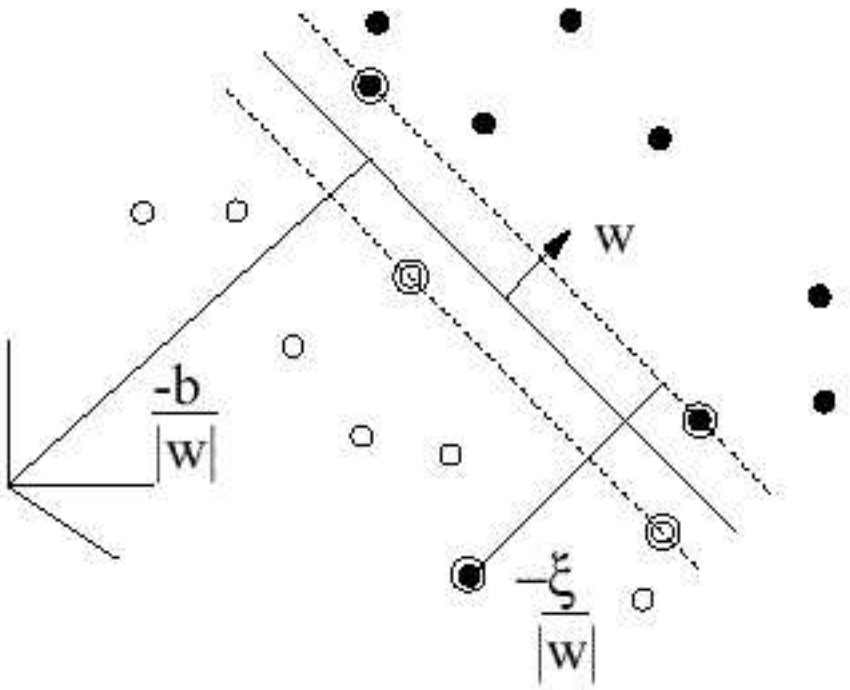

Figure 3.1: Linear Separation Plane for the non-separable case. $b = \gamma$. Courtesy of [5]

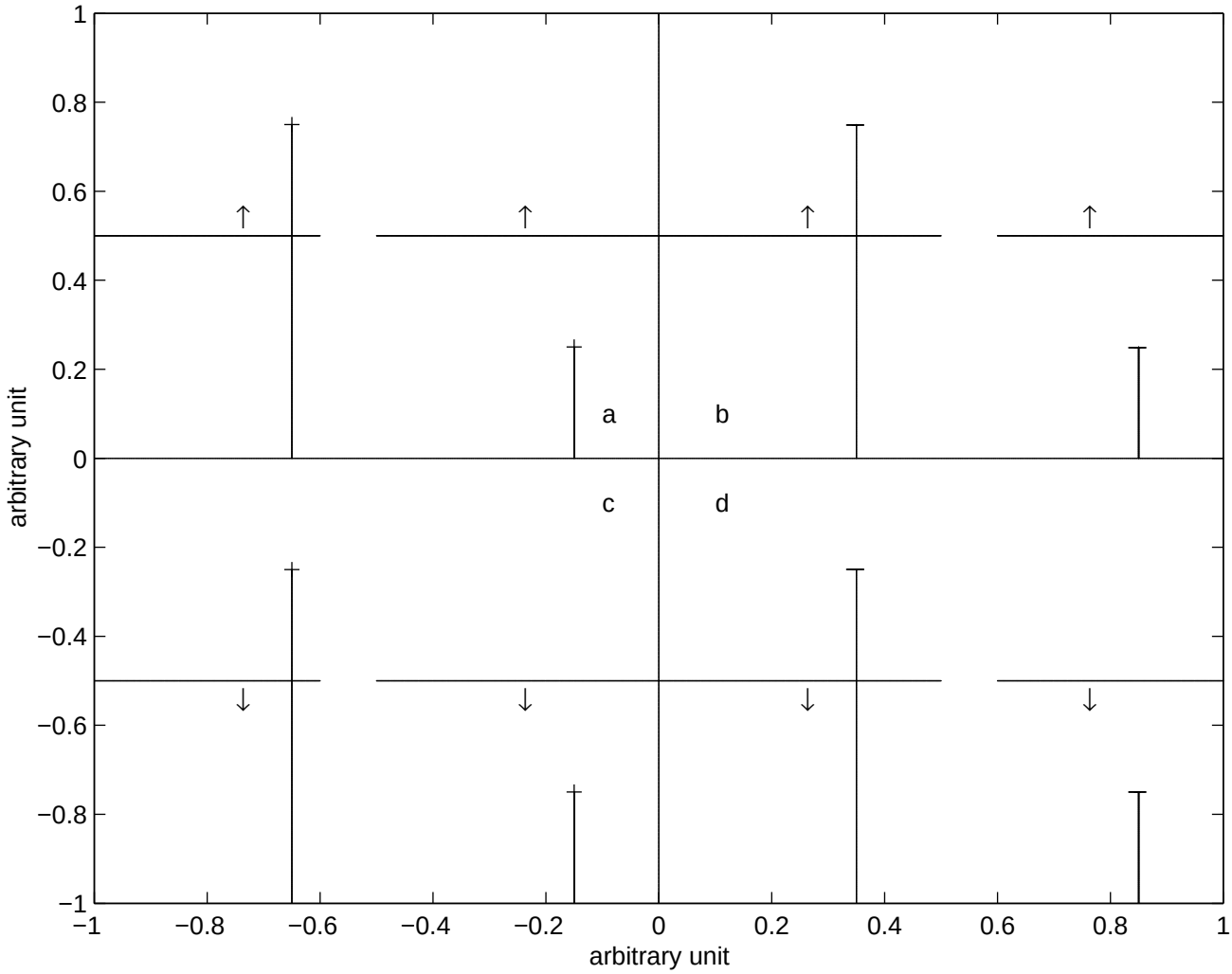

Figure 3.2: Four possible cases about w_μ and y for upper limit shown in the four panels, a-d, corresponding to Table 3.1. In each panel, the μth component of the data, (x, y), may be above (left) or below (right) the separation plane. Only in panels a and d does the position relevant to the separation plane make sense. The hypothetical separation plane is always satisfied in panels b and c no matter where the data item is positioned. The arrow in each figure points in a w_μ positive direction. The vertical line below the data + or - indicates that it is an upper limit.

	w_μ	y	training feasibility
a	+	+	yes
b	+	-	no
c	-	+	no
d	-	-	yes

Table 3.1: Training feasibility for the four cases of w_μ direction and class y. See also Fig. 3.2a-d) for illustration.

sification if $\hat{x}^p$ was below the separation plane. That is to say, the separation plane does matter in terms of classification in this case. In other words, the hypothesis that the plane predicts the point correctly depends on the upper limit at the component, $\hat{x}_\mu$. Therefore, training needs to take this data into account;

b) when w_μ is positive and y is negative, the "+1" class is expected to be above the separation plane, and the "-1" class below the separation plane along the μth component. On the other hand, this point has a negative class, and it has an upper limit along the μth component. It does not matter whether the upper limit point, $\hat{x}_\mu$, is above the separation plane or not; the point may not contradict the hypothetical classification according to the current separation plane. In other words, one cannot reject the hypothesis that the plane correctly predicts the point class anyway;

c) when w_μ is negative and y is positive, the "+1" class is expected to be below the separation plane along the μth component. Wherever $\hat{x}_\mu$ is, $x_\mu \leq \hat{x}_\mu$ may not be above the plane, and so the point may not contradict the separation that is expected with the prediction using the separation plane along its μth component. In other words, the hypothesis that the separation plane correctly predicts the point class cannot be rejected;

d) when w_μ is negative and y is negative, the "+1" class is expected to be below the separation

plane, and the "-1" class is expected to be above the separation plane. x is expected to be above the plane. Because of the existence of the $\hat{x}_\mu$ upper limit, however, the point may or may not be above the separation plane. Moreover, the point is absolutely below the separation plane when $\hat{x}_\mu$ is so. That is, the hypothesis that the separation plane correctly predicts the point class depends on the upper limit, $\hat{x}_\mu$. Classification training must take this point into consideration.

This generalization of the four cases provides the necessary and sufficient condition, $w_\mu y > 0$. Here, when the point falls on the plane, it is regarded as a positive classification; a parallel plane along the μ axis means $w_\mu = 0$. The parallel plane along the μ axis is not sensitive to the location of x_μ. So, the condition specified in Theorem 1 is also correct in both of these two special cases. This completes the proof of Theorem 1.

Theorem 2: If a point, $p = (x, y)$, containing upper (lower) limits satisfies the condition in Theorem 1, its upper (lower) limit, $\hat{x}_\mu$ ($\check{x}_\mu$), of the μth component can be used to replace x_μ in SVMs training.

Theorem 2 provides the basis for the iterative operation of the implementation of training algorithms in ULEDS-SVMs. To prove the theorem, the upper limit case is considered. The lower limit case can be similarly proved. Two cases need to be considered: $w_\mu > 0$ and $y = +1$; and $w_\mu < 0$ and $y = -1$.

1) $w_\mu > 0$ and $y = +1$: First, it is necessary that every $\hat{x}$ of the point with a "+1" class satisfies the classification predicted correctly using the separation plane, if x can be so classified. It also holds that if the plane misclassifies $\hat{x}_\mu$, then it also misclassifies x. To prove this idea, assume that $\hat{x}_\mu$ does not satisfy the separation hypothesis. That is, at least one component, μ,

exists such that $\hat{x}_\mu$ is below the separation plane along its μth component. Then, by definition, $x_\mu \leq \hat{x}_\mu$ must also be below the separation plane along the μth component. So, using $\hat{x}$ to replace x is necessary. Second, it is sufficient to use $\hat{x}_\mu$ as the μth component value to replace x in training. To prove this, it is recognized that if a point at $\hat{x}$ is correctly classified, it is above the separation plane along its μ component. This is an upper limit point. The actual data point could be anywhere below this upper limit, but its exact location is unknown. However, according to the definition of the hypothesis of plane separation, the hypothesis does not contradict the classification of point, x, given the knowledge available.

2) By using a similar proof, it can be shown that $\hat{x}_\mu$ can replace x_μ for the point when $w_\mu < 0$ and $y = -1$.

It can be proven that the lower limit, $\check{x}_\mu$, along every component can replace x_μ in the lower limit case. This completes the proof of Theorem 2.

Analogous to equations 3.1 and 3.2, the primal Lagrange function, including data with upper and lower limits, is:

$$\min_{(w,\gamma,\xi)\in R^{n+1+m}} \nu e' M\xi + \frac{1}{2} w'w \ \text{ s.t. } M(D(Aw - e\gamma) + \xi - e) \geq 0,\ M\xi \geq 0 \tag{3.3}$$

and the dual to the Lagrange function can be written as:

$$\min_{\alpha\in R^m} \frac{1}{2}\alpha' MDAA'DM\alpha - e'\alpha \ \text{ s.t. } e'DM\alpha = 0,\ \ 0 \leq \alpha \leq \nu e. \tag{3.4}$$

where A is the matrix for entities, including data with upper and lower limits, and M is a

diagonal matrix in $R^{m \times m}$ with diagonal elements. $M_{ii} = 0$ if $w_\mu y^i < 0$ for any upper limit or if it is $w_\mu y^i > 0$ for any lower limit component, and $M_{ii} = 1$ otherwise. Here, i corresponds to the index of the item, while a Greek subscript corresponds to an attribute. The problem, therefore, becomes a typical linear SVM problem. Available algorithms concerning SVMs are applicable to treating data with upper and lower limit cases with such an embracement. They can be adopted with the simple modification of keeping or dropping the entity with the upper or lower limit under the conditions set forth in Theorem 1. If the entity is kept, it replaces the corresponding value, according to Theorem 2.

In training, there are two possible approaches: training the whole data set with an M mask, as in the above equations, or training only the subset of the data which have the corresponding non-zero mask value. The approach that is chosen will affect the time complexity and can be determined by the fraction of the number of data entities with limits or errors (which will be discussed later).

3.2 Test

According to Theorem 1, if $w_\mu y \leq 0$ for an upper limit or $w_\mu y \geq 0$ for a lower limit at any attribute, μ, the classification is always correct. So, the test result is true. On the other hand, if $w_\mu y > 0$ in the upper limit case or $w_\mu y < 0$ in the lower limit case, according to Theorem 2, it is necessary to test the upper (lower) limit point, $\hat{x}$ ($\check{x}$), against the separation plane in the same way as one would if it had a definite value. That is, if $(w'\hat{x} - \gamma)y \geq 0$ or $(w'\check{x} - \gamma)y \geq 0$, then the point is correctly classified, and the test result is true. Otherwise, it is misclassified, and the

test result is false.

First, this method utilizes the class, y. And, it is applicable only for data with a limit case. Alternatively, one can use the prediction method described in the next section and then determine if the prediction is the same as the given class. This utilizes the concept of predictability (deferred to the next section). The current approach is nonetheless useful for the limit data test because of the reduction in computing time.

3.3 Prediction

Unlike the use of conventional SVMs to predict definite data entities, predicting an entity with an upper (lower) limit may result in "+1", "-1" or "unpredictable" which makes it impossible to determine its class based on the uncertain attribute values if some conditions are satisfied. Here, an intersection concept is introduced in order to cast the prediction feasibility.

Definition: for an upper (lower) limit case, if the upper (lower) limit point, $\hat{x}$ ($\check{x}$), is above (below) the separation plane along its μth component, then the actual entity value at the μth component, x_μ, can be on either side of the separation plane. One says, then, that the upper (lower) limit, $\hat{x}_\mu$ ($\check{x}_\mu$), intersects with the separation plane. If none of the upper (lower) limits intersect with the plane, then one says that there is no intersection.

Extending the intersection concept further to data with error, one says that an entity has a component with an intersection if both its upper and lower limits at the corresponding component intersect with a separation plane.

If an upper (lower) limit or data with error intersect with the separation plane, the class of

the corresponding entity is not predictable. If an entity does not intersect with the separation plane, then its class is predictable, and its class is as predicted for its upper (lower) limit point for limit data or anywhere within the region of error for error data. A centroid value is chosen in the error case.

For a compact expression of the next Theorem, the code, $U \in R^n$, is denoted as a vector of the limits, where $U_\mu = 1$ represents the upper limit, -1 represents the lower limit and 0 represents the definite value of an attribute of the data, (x, y), on the μth component.

Theorem 3: The intersection and its inherent nonpredictability of a data point, (x, y), is determined by the existence of a component of μ such that $(w'x-\gamma) = 0$ or $(w'x-\gamma)w_\mu U_\mu > 0$ for upper or lower limit data, and $(w'_\mu \hat{x} - \gamma)(w'_\mu \check{x} - \gamma) \leq 0$ for error data where x is the vector of a data point, (w, γ) the vector of the separation plane, U_μ the code of limits on the μth component, ${}_\mu\hat{x}$ the vector of the point whose μth component is set at its upper limit while the other components are set at their centroid values, and ${}_\mu\check{x}$ the vector of the point whose μth component is set at its lower limit while the other components are set at their centroid values.

This explanation leads to the proof of Theorem 3 from the geometric relationship of a data point with limits or errors with respect to the separation plane. The condition of $(w'x - \gamma) = 0$ indicates that the upper or lower limit is right at the separation plane. Since this is a limit point, this means that it intersects with the plane. In the case where the actual data point falls on the separation plane, one cannot tell to which class it belongs. So, it is appropriate that the data point is included in the intersection case. In order to focus now on the other conditions, four cases need to be considered for limits:

1) $w'x - \gamma > 0$, $w_\mu > 0$ and $U_\mu = 1$ indicate that the μth component of the separation plane normal points upward, and the limit point is above the separation plane along its μth component. But the point is an upper limit point, meaning that there is an intersection.

2) Similarly, $w'x - \gamma > 0$, $w_\mu < 0$ and $U_\mu = -1$ indicate that the μth component of the plane normal points downward, and the limit point is below the separation plane along its μth component. But this point is a lower limit point, meaning that there is an intersection.

3) $w'x - \gamma < 0$ and $w_\mu > 0$ and $U_\mu = -1$ indicate that the μth component of the plane normal points upward, and the limit point is below the separation plane along its μth component. But this point is a lower limit point, meaning that there is an intersection.

4) $w'x - \gamma < 0$, $w_\mu < 0$ and $U_\mu = +1$ indicate that the μth component of the plane normal points downward, and the limit point is above the separation plane along its μth component. But the point is an upper limit point, meaning that there is an intersection.

All four of these cases have exhausted the intersection cases where there is a limit in their attributes. The combined formula satisfies $(w'x - \gamma)w_\mu U_\mu > 0$. For error data, the following case must be considered:

5) $(w'_\mu \hat{x} - \gamma)(w'_\mu \check{x} - \gamma) \leq 0$ means that the upper and lower boundaries are separated on either side of the separation plane no matter which w_μ direction the normal is, indicating that there is an intersection. This proves Theorem 3.

With Theorems 1 and 3, one can immediately obtain:

Corollary: a datum with error is unpredictable if and only if it intersects with the separation plane.

This is because if it intersects, one of the boundaries of the error must satisfy the Theorem 1 conditions. If it does not intersect, the boundaries of the error are on one side of the separation plane.

It is worth pointing out that the feasibility for training addressed in Theorem 1 is different from the feasibility of prediction addressed in Theorem 3 for limit cases. An item from a data set may be trainable but not predictable, as shown, e.g., on the left plot in Fig. 3.2a. Similarly, it may be predictable but not trainable, as shown, e.g., on the right plot in Fig. 3.2b. For data with error, prediction feasibility leads to training feasibility and vice versa. This is because 1) if the boundaries of an error along one attribute component are on the same side of the separation plane in respect to this attribute, then there is no intersection. Training is feasible. 2) If both boundaries of an error along one attribute component are on separate sides of the separation plane in respect to this attribute, as expressed in the formula in Theorem 3 for error data, then there is intersection. No matter what class the data item is in, and no matter what direction w_μ goes, there is at least one boundary satisfying the Theorem 1 condition. Geometrically, this point satisfies the hypothetical separation plane already, and it does not suggest any more information about which way the separation plane would go during the current iteration for training.

Chapter 4

Algorithms

There are many implementations for SVMs, e.g., managing large data sets, such as SVM^{Light} [15], based on [21]'s segmentation scheme, or addressing the issues for overcoming overfit, such as the smooth method [18], etc. In chapter 8, a detailed review of the SVMs and some of their recent developments is provided. The purpose of this section is not to provide a benchmark, but rather, discuss general algorithms which can be applied to the implementation of most SVMs in domains that contain data with limits and errors, based on the principles discussed in the previous chapter.

First, the algorithms that handle data with upper and lower limits for training, testing and prediction will be described. This idea is then extended to apply to data with errors. These algorithms utilize available SVMs by invoking them directly in training; this is highly portable from a software engineering point of view. Conventional SVMs do not need to be modified in this scenario. The portion of the algorithm that handles the limit or error data is used for the

preparation of data by following the conditions that satisfy the trainability, as described above. Then, the prepared data are fed to conventional SVMs for training.

When both limits and errors exist, the above scenario can also be used. However, with a slight modification of conventional SVMs, the efficiency of the algorithms can be improved in cases where a large fraction of limit or error data exists. This improvement can be achieved in each iteration by preparing a mask for the data in order to satisfy the trainability conditions, followed immediately by one iteration of the SVM training. An alternative approach may also be considered to increase the SVM training iteration a few more steps, but this is not necessary until the convergence of the computation process. Along with this approach, the modification of conventional SVMs serves simply to reserve the Lagrangian multipliers from the last iteration and use them as the initial Lagrangian multipliers in the next iteration. A pseudocode sample for such a scenario is provided at the end of this chapter.

4.1 Upper and Lower Limit Case

4.1.1 Training

Let D denote the set of all entities with the non-empty subset of D_P which contains only definite data and the subset of entities, D_U, with upper or lower limits for at least one attribute of each entity. Let U denote the set of code vectors of the attributes of the upper or lower limit of each entity in D_U, where 1 represents the upper limit, -1 represents the lower limit and 0 represents the definite value code. Let T represent the subset of D_U which contains trainable

entities and F which contains untrainable entities with respect to a separation plane. Let a working set $W = P \cup T$. And let T_{cond} denote the termination condition.

Algorithm 4.1.1: TRAINING(D_P, D_U, U, T_{cond})

$Initialize \quad (w, \gamma)$

while $termination \quad condition \quad unsatisfied$

do

$T = empty$

for each $entity \quad p \in D_U \quad u \in U$

do

if $Trainable(p, u, w) \quad == TRUE$

then $T \leftarrow$p

else $F \leftarrow$p

$W = D_P \cup T$

comment: invoke conventional training

$call \quad convTraining(W, w, \gamma, T_{cond})$

if $Wnonequal \quad to \quad previousW$

then $terminationunsatisfied$

$return(w, \gamma)$

$Function \quad Trainable(p, u, w)$ **comment:** $p = (x, y)$, y is the class of p

for $\mu \leftarrow every \quad upper/lower \quad limit \quad component$

do

if $w_\mu * y * u_\mu < 0$

then $return \quad false$

$return \quad true$

4.1.2 Test

In addition to testing according to conventional methods, which do not have limits, one must also consider the sign of $w_\mu y$ which differs from conventional testing. If the attribute of an entity is < 0 for an upper limit or > 0 for a lower limit case, testing for this entity is always true. Otherwise, it is the same as $(w'x - \gamma)y \geq 0$ to determine the correct classification. The notation, $u \in U$, as the code vector of the limit for an entity is used as the input parameter.

Algorithm 4.1.2: TEST(x, u, y, w, γ)

comment: x - vector containing the attribute values for an entity

comment: y - the class of the attribute

comment: u - vector of entity attribute code: 1/-1/0 - upper/lower/no t limit

comment: w, γ - separation plane direction and position vector/scalar

comment: RETURN - true for "+1", false for "-1" class

for $\mu \leftarrow upper/lower \quad limit \quad components$
  do **if** $w_\mu * y * u_\mu \leq 0$
    then $return \quad true$

if $(w'x - \gamma) * y \geq 0$
  then $return \quad true$
  else
  then $return \quad false$

4.1.3 Prediction

Predictability depends on whether the upper or lower limit point intersects with the separation plane. The function of intersect, Lmt, utilizes Theorem 3. If there is no intersection, prediction is made in the same manner as that of data without limits in the main program.

Algorithm 4.1.3: PREDICT(x, u, w, γ)

comment: x - vector of entity attributes including limits

comment: u - vector of entity attribute code: 1/-1/0 - upper/lower/no limit

comment: w, γ - separation plane direction and position vector/scalar

comment: RETURN - class "+1", "-1", "unpredictable"

if $intersectLmt(x, u, w, \gamma) == true$

then $return \quad unpredictable$

else

if $(w'x - \gamma) \geq 0$

then $return \quad +1$

else if $(w'x - \gamma) < 0$

then $return \quad -1$

$Function \quad intersectLmt(x, u, w, \gamma)$

$f = w'x - \gamma$

if $f = 0 \quad AND \quad there \quad is \quad any \quad limit \quad in \quad u$

then $return \quad true$

for each $\mu \leftarrow upper/lower \quad limit \quad components$

do **if** $(f * w_\mu * u_\mu) > 0$ **comment:** u_μ - μth element of u

then $return \quad true$

$return \quad false$

4.2 Data With Error

4.2.1 Training

Similar to data with upper and lower limits, data with error can be treated the same as if both upper and lower limits, $\hat{x}_\mu > \check{x}_\mu$, are found along the same component. A hyper diamond box formed by the limits surrounding the centroid value of the data point may intersect with the hyperplane which separates the classes, or it may fall to one side of the plane. When there is intersection, this data point is not trainable. Its classification is always true in testing. Because it can be either class, it is nonpredictable. On the other hand, if there is no intersection, training, testing and prediction may be undertaken in the same manner as using only the centroid value of the data point. Conventional SVM algorithms are resumed using these values.

Here, the notation is extended into a space so as to cast the content of the centroid value, the upper and lower boundaries, in case there is an error. Let D denote the set of entities with three-dimensional space, $R^{m\times n\times 3}$. The first two dimensions are the same as the conventional event matrix, but the third is an extension of its centroid value, the upper limit and lower limit values of the corresponding event matrix. If data at an attribute have definite value, the centroid, upper and lower limits are all equal to the definite value. If data at an attribute are at an upper or lower limit, all three values are equal to the corresponding limit. A working set is W in $R^{m\times n}$. Further, the symbols, ${}_\mu\hat{x}$ and ${}_\mu\check{x}$, denote, as previously stated, an entity with attributes that are all set at the centroid value, except along the μth component which is set at its upper and lower boundaries, respectively. As opposed to training data with limits, here the trainable

function is not used. Rather, the predictability Err function is used.

Algorithm 4.2.1: TRAININGERR(D, T_{cond})

$Initialize \quad (w, \gamma)$

while $termination \quad condition \quad unsatisfied$

do

$W = empty$

for each $entity \quad p \in D$

do

if $PredictabilityErr(p, w, \gamma) \quad == TRUE$

then $W \leftarrow$p$'s \quad centroid \quad value$

comment: invoke conventional training

$call \quad convTraining(W, w, \gamma, T_{cond})$

if $W \quad equals \quad to \quad previous \quad W$

then $termination \quad satisfied$

$return(w, \gamma)$

$Function \quad PredictabilityErr(p, w, \gamma)$ **comment:** $p = (x, y)$

for each $attribute \quad \mu$

do

if $({}_{\mu}\hat{x} * w - \gamma)({}_{\mu}\check{x} * w - \gamma) < 0$

then $return \quad false$

$return \quad true$

4.2.2 Testing

When an entity is not predictable, there is an intersection between the hyper diamond error box and the separation plane. This is set true in testing. Otherwise, the test is the same as that performed at the centroid value of the hyper diamond error box. The algorithm is as follows:

Algorithm 4.2.2: TESTERR(x, y, w, γ)

comment: x - the attribute of p

comment: y - the class of p. p is a short representation of x and y

comment: w, γ - separation plane normal and distance

if $PredictabilityErr(p, w, \gamma) == FALSE$

then $return \quad true$

else if $(xw - \gamma)y > 0$

then $return \quad true$

else $return \quad false$

4.2.3 Predicting

Similarly, when a point has an error box that intersects with the hyperplane, it is unpredictable. Otherwise, prediction is possible by using the attribute values of the centroid point with respect to a separation plane:

Algorithm 4.2.3: PREDICTERR(x, w, γ)

comment: x - the attributes of an entity

comment: w, γ - separation plane normal and distance

if $PredictabilityErr(p, w, \gamma) = FALSE$

then $return \quad unpredictable$

else if $xw - \gamma > 0$

then $return \quad +1$

else $return \quad -1$

4.3 ULEDS-SVMs

The algorithms provided in the previous two sections treat data, including those with upper and lower limits or with errors, respectively. When data have both limits and errors mixed in the data set, the scenario used above can also be used. However, this section provides an alternative algorithm to treat the training problem by masking out the untrainable items of data and training the remaining entities within the same iteration loop. Next is the pseudocode for this algorithm, ULEDS-SVMs. Here, the subsets are denoted as follows: $D \subset R^{m \times n+1 \times 3}$ is

the set of data points with lower limits, definite values, and upper limits; $U \subset R^{m \times n}$ is the set for the codes 1, -1, 0 and 2 that correspond to the upper limits, lower limits, error and definite values, respectively; $C \subset R^m$ is the set of classes of the items; (w, γ) is the separation plane; Op is the execution code for training, testing or prediction; and T_{cond} is the termination condition. The α parameter in conventional SVMs is a vector for Lagrangian multipliers. Modification of conventional SVMs can be restricted for output α after each convSVMTraining iteration, and then input α as its initial parameter for training during the next iteration rather than initiating α automatically within convSVMTraining. Iterate within convSVMTraining for training only once or a few times; outside the convSVMTraining, use diag(mask)$*\alpha$ to replace the original α parameters. When there is both "work" set stability and convSVMTraining convergence, training stops at the separation plane, (w, γ).

Algorithm 4.3.1 - ULEDS-SVMs

```
ULEDS_SVMs(D, U, C, w, gamma, Op, Tcond)
// assert: D, U, C are in order of their corresponding attributes
  init d_ret   // common/static vector in Rˆn for attributes of one item
  m = size(C)  // number of items in D
  init d                // init matrix for one item in D, with 3 frame
  init u                // init vector for one item in U
  init c                // init scalar for one item about class in  Rˆ1

  // branch according to Op - training, testing or predicting
  if Op == training       // begin the training branch for ULED
    init w,gamma
    init mask             // init mask vector,
    init Work             // init working set, as a subset of Rˆ{m n+1}

    init iterT = false    // init termination condition in iteration
    while iterT == false
      Work = empty
      // prepare the set by rejecting untrainable items in training
      for i <- 1:m
        // extract one item from the dataset
        (d, u, c) <- extract the ith items from (D, U, C)

        // check out the trainability
        if trainPredictability_ULED(d, u, c, w, gamma, Op) == true
          mask <- 1
        else
          mask <- 0
        Work <- (d_ret, c) // item was put in the working set
      endfor

      alpha = diag(mask) * alpha // mask out the non-trainable item

      // invoke conventional SVMs training routine
      // with a little modification
      call convSVMTraining(Work, w, gamma, alpha, Tcond)

      if Work does not change in consequent iteration and Tcond satisfied
        iterT =  true
    endwhile
    return -1    // return the -1 code after successful training

  elseif Op == testing    // begin testing branch
    init count = 0  // init the number of correct classification in test
```

```
  init Right      // init the set in R^{m n+1} classified correctly
  init Wrong      // init the set in R^{m n+1} classified incorrectly

  for i <- 1:m    // test the item one by one
    // extract one items from the dataset
    (d, u, c) <- extract the ith items from (D, U, C)

    // check if predictable or not
    if trainPredictability_ULED(d, u, c, w, gamma, Op) == true
      // predictable, then test as for the normal predication case
      if c * (d_ret * w - gamma ) >= 0  // classified correctly
        count++
        Right <- (d_ret, c) // output into Right class set
      else                                    // misclassified
        Wrong <- (d_ret, c)  // output into Wrong class set
    else
      // unpredictable, then set as classified right
      count++
      Right <- (d_ret, c)   // output into Right class set
  endfor
  return count       // return the correct classification number

elseif Op = predicting            // begin predicting branch

  for i <- 1:m     // predict the items one by one
    // extract one items from the dataset (d_class is unnecessary)
    (d, u, c) <- extract the ith items from (D, U, C)

    if trainPredictability_ULED(d, u, c, w, gamma, Op) == true
      // predictable, then predict as definite case
      if (d_ret * w - gamma ) > 0
        Dclass(i) = 1             // +1 class
      elseif (d_ret * w - gamma ) < 0
        Dclass(i) = -1            // -1 class
      elseif (d_ret * w - gamma ) = 0
        Dclass(i) = 0             // right on separation plane
    else
      // unpredictable, then all set as 0 class
      C(i) = 0
  endfor
  return -3                       // return the prediction code -3
```

```
function trainPredictability_ULED(d, u, c, w, gamma, Op)
  // function for the trainability and predictability evaluation
  n = size(u)                        // number of attributes
  for mu <- 1:n
    d_ret(mu) = d(mu,2)              // assign the centroid value to d_ret
  endfor

  for mu <- 1:n                      // loop run through the attributes
    if u(mu) == 0
      // case of attribute with error
      if intersectCompErr(mu, d(mu,1), d(mu,3), w, gamma) == true
        return false
    elseif abs(u(mu)) == 1
      // case of attribute with limit (u(mu) = 1 or -1)
      if Op == training
        if trainabilityCompLmt(u(mu), c, w(mu)) == false
          return false
      if Op == testing or Op == predicting
        if intersectCompLmt(mu, u(mu), w, gamma) == true
          return false
  endfor
  return true

function intersectCompErr(mu, d_low, d_up, w, gamma)
  xup <- d_ret       // copy the vector for the centroid value into xup
  xup(mu) <- d_up    // replace xup's mu'th element by d_up
  xlow <- d_ret      // copy the vector for the centroid value into xlow
  xlow(mu) <- d_low  // replace xup's mu'th element by d_low
  if (xup * w - gamma) * (xlow * w - gamma) <= 0
    return true      // one of theorem 3 intersection conditions satisfied
  else
    return false     // no intersection along this component

function intersectCompLmt(mu, u_mu, w, gamma)
  f = d_ret * w - gamma          // objective function
  if f == 0 return true          // one end of the limit right on the plane
  if f * w(mu) * u_mu > 0
    return true                  // one of theorem 3 conditions satisfied
  return false

function trainabilityCompLmt(u_mu, w_mu, c)
  if w_mu * u_mu * c <= 0
    return false                 // theorem 1 condition violated
  else
    return true                  // theorem 1 condition satisfied
```

Chapter 5

Chandra and Hubble Deep Field North (CHDF-N) dataset

Multiband observations for Hubble Deep Field (HDF) and Chandra Deep Field(CDF) North are used as the test bed in this study. This sky direction is used to avoid, as much as possible, the stars from our galaxy and interstellar media so that observations can be as deep as possible. X-ray observations using Chandra are from [4]. This reference contains, in its original catalogue, 370 sources which were discovered using an average of about 1 Ms exposure observation. A more recent result based on a 2 Ms exposure was also available [1] at the time this work was completed. However, because the new catalogue contains a large fraction of the total sources that contain observational results that are not publicly available at wavebands other than X-ray for magnitudes similar to those used here, the 1 Ms catalogue is still used for this work. Three attributes were selected from [4]. These are fluxes of the full band, FB, the soft band, SB, and

the hard band, HB.

[2] provides a source catalogue with magnitudes at several energy bands other than the X-ray band of [4]'s X-ray observations. The source attributes include the magnitudes from submillimeter, 20 cm radio, infrared, and several optical filter photometric observations. The 20 cm radio survey observation contains a much less positive detection than upper limits. Even though the method developed in this paper targets training with toleration regarding the existence of upper and lower limits and errors, this attribute was eliminated for the purpose of the SVM training, testing and prediction. That is, in this research, the astronomical aspects of this work are not emphasized. The field for submillimeter observations is much smaller than for other observations. Therefore, there is a large number of missing values for the submillimeter attribute. In addition, even within the observed area, there are many upper limits, or values with large errors. So, here the submillimeter attribute is dropped, too, for the same reason that the radio attribute was eliminated. The attributes now used include the infrared magnitude, HK', and optical magnitudes with filters of z', I, R, V and B, with corresponding colors from red to blue. The relationship between flux and magnitude is [**?**] $Mag = -48.6 - 2.5log(F)$ where Mag is the magnitude of an object, and F is the flux of the source. As defined, the brighter a source, the less value its magnitude. So, for source saturation in a photometric observation, a lower limit of its brightness may be obtained which corresponds to an upper limit in magnitude. On the other hand, a faint source with a negative detection on some wavelengths results in an upper limit of brightness which corresponds to a lower limit in magnitude. Magnitude limits for some of the sources with saturation or a faint limit are treated this way by referring to the

survey limits described in [2].

For classification of the objects, one of the most important methods is to categorize objects into stars, galaxies, active galactic nuclei (AGNs), etc. The stars, galaxies and AGNs may be further categorized according to subclasses. [13] identified and compiled, among 82 early Chandra deep field X-ray sources, 32 sources for the classes of the objects. One of the sources, (#39 in [13]), appears in neither [4]'s X-ray nor [2]'s optical catalogues. But, it is included in [1]'s 2 Ms observation catalogue. Because there is a lack of attribute information for every energy band except for X-rays, this source has been eliminated from the analysis.

Among the remaining 31 sources, there are 2 stars, 8 low luminous galaxies (LL), 7 high luminous galaxies (HL), 8 broad line AGNs (BL), and 7 narrow line AGNs (NL). Given the sample size of the identified objects, the low and high luminous galaxies are combined into one class (-1), and the broad and narrow line AGNs into another (+1). Stars can be classified relatively easily from multiband observations.

[2] also made spectroscopic observations of the bright optical counterparts of the 370 X-ray sources. Although an atlas for the spectrum is provided, it has not been classified according to the conventional types of astronomical objects, except for 5 stars. The intention here is to integrate the data uniformly, and because the classification of stars is not a major issue of astronomical interest here since the main focus is the classification of extragalactic objects, and also because the classification for the known extragalactic objects is mainly in the subset that does not interact with the star class, no further effort is made to include [2]'s stars in the data set below.

The composed data set for those objects with known identifications is shown in Table 5.1 with columns 1-2: the source sequence number in this Table and the source number in [4] [2]; columns 3-4: the source coordinates, Ra and Dec, from [2]. These coordinate attributes were not used in training. Including these attributes here is for convenience of identification only; columns 5-10: magnitudes of HK', z', I, R, V, and B from [2]; columns 11-13: X-ray flux in a unit of 10^{-15} ergs/cm^2/s from [4]; columns 14-15: the source identification from [13] and its digitized class, identified by combining galaxies as -1 and AGNs as +1, respectively, as described above. The brackets "[ ]" quote the upper limit of a magnitude where saturation was encountered due to a bright source. This includes 2 stars (No. 6 & No. 31) and 1 galaxy (No. 23) at some optical wavelength with filter observations. In addition, 4 galaxies (No. 10, 15, 23, 25) have upper limits at the HB X-ray measurements, as marked by "<". Among these, source No. 23 has both optical saturation and upper limits at the HB X-ray band.

It is reasonable to use a log scale for FB, SB and HB. But here in Table 5.1, the energy flux is listed as the original linear scale, as in [4]. In the next chapter, it is log-scaled with 2.5*log(Flux) in training in order to scale up to match the scale in optical observations.

Without an identification constraint, there are 274 sources out of [4]'s original 370 sources which satisfy the criteria. That is, the sources with the missing attributes were eliminated. This information is compiled and presented in Table 9.1 in Appendix A where the values above the slash indicate the magnitude and fluxes of the corresponding filter or energy bands, and the values below the slash indicate the boundary code; this is a little different from U described in the previous chapter because there is no error case here, where 1 is the upper limit, -1 is the

lower limit and 0 is the definite value.

No.	Seq	Ra	Dec	HK'	z'	I	R	V	B	FB	SB	HB	ID	class
1	86	189.071044	62.16980	18.8	21.5	21.6	22.9	23.8	24.5	4.54	1.41	3.05	HL	-1
2	88	189.075088	62.27650	18.0	20.3	20.1	21.0	21.6	22.3	15.11	4.61	10.69	NL	+1
3	89	189.077468	62.18750	18.3	20.9	20.8	21.3	21.7	21.9	16.08	6.75	9.21	BL	+1
4	98	189.088409	62.18578	18.7	21.7	21.9	23.4	24.6	25.5	3.04	0.08	3.03	HL	-1
5	101	189.095626	62.25744	18.6	20.2	20.0	20.3	20.3	20.5	16.33	6.07	10.37	QSO	+1
6	108	189.105865	62.23466	14.4	[18.0]	[18.0]	[18.0]	[18.0]	20.3	1.62	1.12	0.24	M4Star	*
7	117	189.121338	62.17942	18.8	22.2	22.4	23.8	24.8	26.0	2.31	0.16	2.22	NL	+1
8	118	189.122253	62.27050	18.4	21.0	20.9	22.3	23.2	24.0	2.27	0.06	2.36	NL	+1
9	130	189.139526	62.23836	21.3	26.0	25.1	25.0	25.3	26.4	1.00	0.42	0.61	BL	+1
10	134	189.143600	62.20359	16.4	19.3	19.2	19.9	20.9	22.1	0.43	0.23	< 0.27	SB/LL	-1
11	141	189.147079	62.18608	18.0	20.5	20.4	21.3	22.4	23.8	0.95	0.17	0.76	LL	-1
12	144	189.152679	62.22969	18.9	21.1	20.9	21.5	22.0	22.6	10.09	4.43	5.53	BL	+1
13	145	189.153122	62.19889	19.1	21.5	21.4	22.2	22.9	23.5	4.94	1.97	2.90	NL	+1
14	152	189.164917	62.20847	20.6	24.1	23.8	24.5	25.4	26.0	0.69	0.33	0.38	BL	+1
15	160	189.174164	62.19222	18.3	20.0	19.7	20.1	20.3	20.9	0.38	0.13	< 0.26	LL	-1
16	163	189.175903	62.26272	18.1	21.2	21.2	22.4	23.3	24.3	3.20	0.84	2.48	HL	-1
17	171	189.193130	62.23466	18.2	20.9	21.0	22.2	22.8	23.8	22.79	2.80	20.12	BL	+1
18	176	189.200211	62.21919	17.9	20.3	20.2	21.1	22.3	23.7	0.75	0.18	0.63	Ellip/LL	-1
19	190	189.215683	62.20595	19.0	21.4	21.2	21.9	22.5	23.6	2.81	0.28	2.59	QSO	+1
20	193	189.220459	62.24561	16.6	19.1	18.9	19.8	20.7	22.1	0.57	0.29	0.29	LL	-1
21	198	189.231094	62.21983	18.8	21.6	21.9	23.3	24.3	25.5	1.27	0.40	0.89	Ellip/HL	-1
22	203	189.237198	62.21711	19.3	22.4	22.6	23.6	24.2	24.8	0.86	0.36	0.49	Ellip/LL	-1
23	209	189.243362	62.16625	16.0	18.1	[18.0]	[18.0]	19.1	20.1	0.23	0.10	< 0.27	LL	-1
24	212	189.245209	62.24303	17.9	20.6	20.4	21.5	22.2	23.0	7.31	2.30	5.17	BL	+1
25	220	189.258484	62.18956	17.6	19.5	19.2	19.7	20.1	20.9	0.15	0.05	< 0.18	LL	-1
26	223	189.261307	62.26214	17.2	19.5	19.4	20.0	20.5	21.0	24.74	4.30	20.51	post-SB/HL	-1
27	228	189.269196	62.28111	18.1	20.3	20.2	20.7	21.4	22.3	1.72	0.66	1.09	NL	+1
28	237	189.278549	62.28397	18.0	19.2	18.8	19.5	19.6	20.0	31.94	14.17	17.15	BL	+1
29	264	189.307541	62.24014	18.6	21.3	21.2	22.1	23.1	24.3	1.75	0.13	1.69	HL	-1
30	274	189.316452	62.20364	19.0	22.2	22.4	23.7	24.7	25.3	4.40	0.07	4.49	HL	-1
31	302	189.357132	62.28017	16.0	[18.0]	[18.0]	19.7	20.7	22.1	0.52	0.25	< 0.32	M4Star	*

Table 5.1: Table of identified CHDF-N sources with identification for galaxies and AGNs. Two stars from [13] are also included. Galaxies are assigned as -1, and AGNs +1 in its two-class categorization. In ID column the symbols are for abbreviations of HL - high luminous galaxy; LL - low luminous galaxy; BL - broad line AGN; NL - narrow line AGN; QSO - AGN; SB - star burst galaxy; M4Star - M4 type star. See text for details.

Chapter 6

Classification via ULEDS-SVM on CHDF-N Dataset

6.1 Implementation of ULEDS-SVMs Incorporated with LSVM

Chapter 8 will provide a detailed review of the SVMs and their implementation. Many SVM implementations exist that emphasize different aspects. For the purpose of an implementation that extends SVMs into ULEDS-SVMs and applies these ULEDS-SVMs in the CHDF-N data set, the Lagrangian SVM (LSVM) [20] was chosen as the conventional SVM that is most relevant to ULEDS-SVMs. LSVM is simple in implementation within a Matlab environment and is especially suitable to showcase in the classroom.

The code for ULEDS-SVMs used here is written in Matlab language to utilize the matrix handling advantage of Matlab. Otherwise, Matlab tools are limited to plotting the results of

μ	1	2	3	4	5	6	7	8	9	γ
$(\bar{w}, \bar{\gamma})$	0.5120	0.4993	0.3075	-0.3240	0.0052	-0.4486	-0.3593	-0.1657	0.3553	-1.2501
$\bar{x}$	18.3	21.0	20.9	21.8	22.5	23.4	1.27	-0.399	0.854	

Table 6.1: The parameter of the separation plane, $(\bar{w}, \bar{\gamma})$. The averages of the attributes, $\bar{x}$, are also listed here for normalization.

this work. The code is given in Appendix B.

6.2 Classification of the Star Class

Even though the main interest here is in extragalactic objects in the CHDF-N sky direction, it is nonetheless necessary to separate stars from other objects first to demonstrate the capability of the ULEDS-SVMs to separate the extragalactic objects as well as to clean up the data set for extragalactic objection classification. Two stars are designated as one class, and all the rest as another class. If one did not include the four sources with upper limits in the X-ray HB measurement in classification, the stars could also be separated easily from the rest of the sources. However, this separation, (w, γ), might not allow the four sources with X-ray HB upper limits, No. 10, 15, 23 and 25, to be satisfied. Therefore, these four sources are also included in the classification.

For a parameter of ν larger than 1.0 it is found that the two classes are separable. Figure 6.1 shows the separation of the stars from extragalactic objects in terms of the objective function, $f(x) = w * x - \gamma$, in the X-axis versus its actual class in the Y-axis. With the separation plane determined by ULEDS-SVMs, it is seen in Table 6.1 that the most dominant component In this classification is the first attribute, followed by the second, and then the sixth and seventh

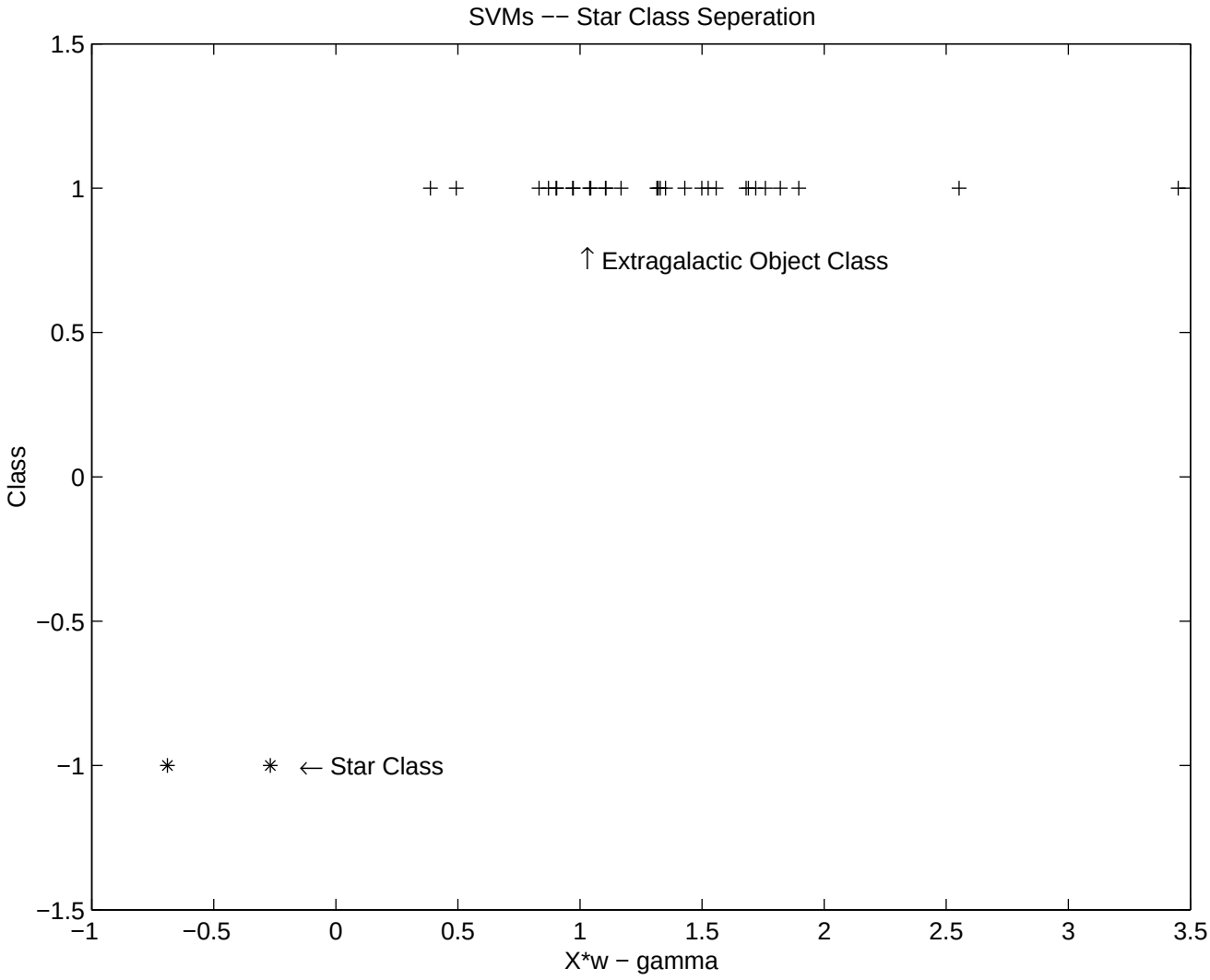

Figure 6.1: Classifi cation of stars from extragalactic objects. The asterisks for stars should be understood as upper limit in the X-axis. The two classes are well separated.

attributes. In Figure 6.2, the data points are plotted in the projection of the $(w1 * x1 + w2 * x2)$ versus $(w6 * x6 + w7 * x7)$ plane, where stars are identified with asterisks. The vertical lines under the asterisks indicate the upper limits at the projected component. The line is $((w1 + w2) * t + \gamma, (w(6) + w(7)) * t)$ and is the parameterized line function derived from the separation plane, (w, γ). It separates the star class from the rest. The four right most points above the line are the upper limit points at HB. It can be seen that without these four data points, separation would tend to be flatter. The addition of the four upper limit data points helps to constrain the separation of the two classes by extending the range of data, and so reducing any ambiguity.

With only the two star class data, which contain the upper limits of the magnitude in some of the optical energy bands, one would not want to exaggerate the results. Rather, this suggests that the separation of the star class from the other classes is feasible. In the following, the stars are excluded from the extragalactic object analysis so that the focus may be on the extragalactic objects only.

6.3 Classification of Extragalactic Objects

The extragalactic object classification for galaxies (-1) and AGNs (+1) was trained, tested and predicted. Extragalactic object classification provides an opportunity to address some of the questions that were not warranted in discussions of star class separation due to the sample size limitation. One concern is the methods of training and testing. Another is the optimization of the ν parameter in the Lagrangian function.

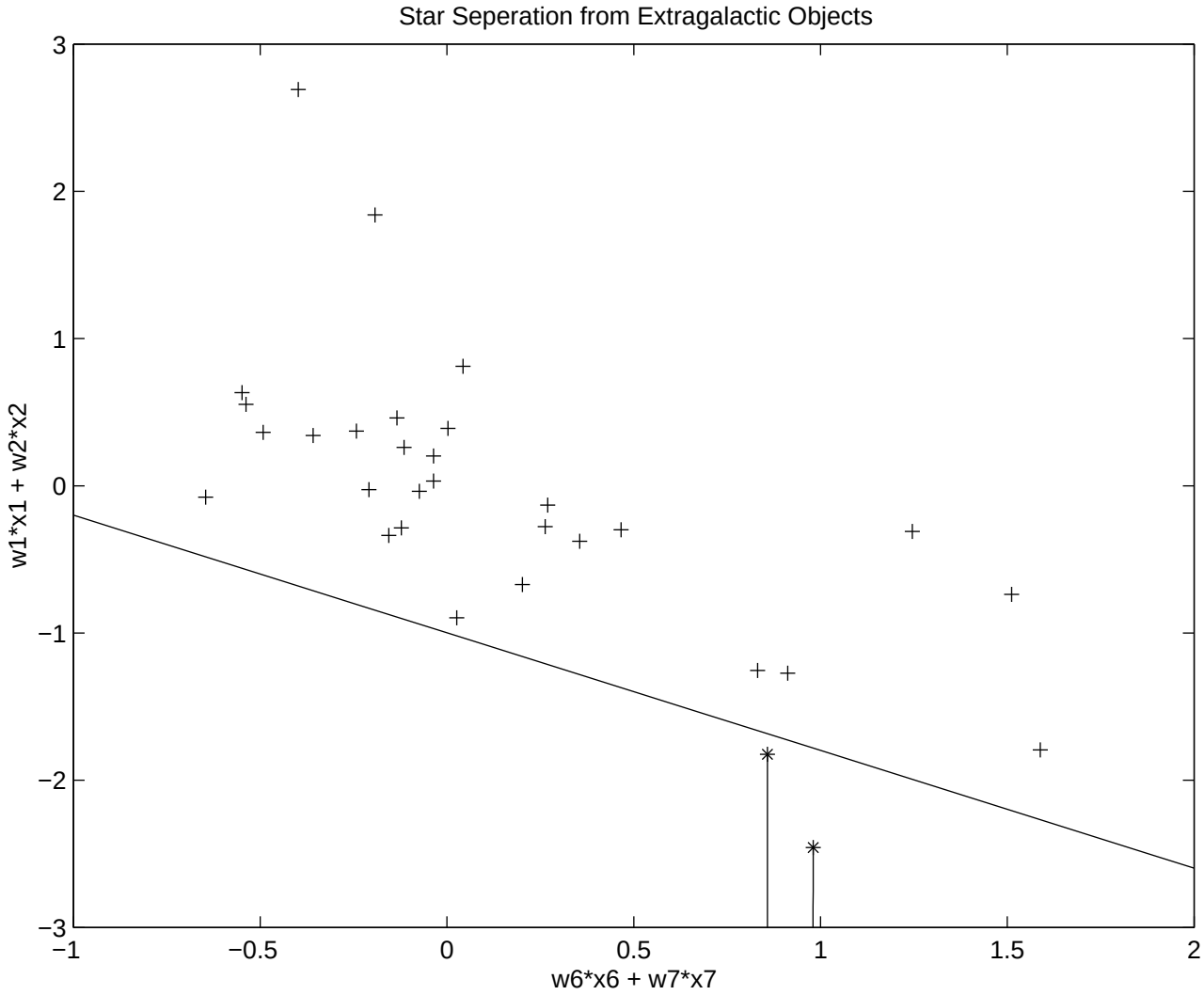

Figure 6.2: A plot of the projection of the star and extragalactic classes on its analogical ”principle component”. Asterisks represent stars, and + signs represent extragalactic objects. Upper limits are given for the stars on the projected component. The four extragalactic objects with upper limits are at the far right end above the line. Without these four upper limit sources, separation can also be made easily, but would tend to be flatter. Adding the four extragalactic objects with upper limits results in better constraint on the separation.

μ	1	2	3	4	5	6	7	8	9	γ
$(\bar{w}, \bar{\gamma})$	0.2064	0.1189	0.0511	-0.0139	-0.0675	-0.0737	0.1377	0.1689	0.1090	-0.0594
$(\delta\bar{w}, \delta\bar{\gamma})$	0.0122	0.0075	0.0090	0.0100	0.0092	0.0092	0.0091	0.0225	0.0147	0.0253
$\bar{x}$	18.568	21.240	21.148	22.052	22.784	23.628	1.3837	-0.3756	1.0335	
$\delta\bar{x}$	0.9433	1.4620	1.4219	1.4669	1.6056	1.7339	1.3471	1.7469	1.3657	
$(\bar{w}_e, \bar{\gamma}_e)$	0.2036	0.1159	0.0508	-0.0063	-0.0609	-0.0750	0.2045	0.2049		0.0
$(\delta\bar{w}_e, \delta\bar{\gamma}_e)$	0.0113	0.0067	0.0079	0.0092	0.0093	0.0101	0.0161	0.0190		0.0237

Table 6.2: The parameter of the separation plane, $(\bar{w}, \bar{\gamma})$ for extragalactic object separation: AGNs and galaxies. The averages of the attributes, $\bar{x}$, are also listed here for normalization. The rows with δ represent the standard deviation of the corresponding parameters. The rows with the subscript, ë", indicate the results after eliminating the HB attribute and source No. 23.

To verify training and testing it is appropriate to use the leave-one-out method due to its limited sample size rather than the three-fold or ten-fold methods usually used. In the leave-one-out method, a sample is withheld from training. This withheld sample is used as a test sample against the separation plane that was obtained from training the remaining samples. The average of the separation plane, $(\bar{w}, \bar{\gamma})$, obtained from training each of the subsets by holding one out, is the final separation plane. Training accuracy is measured by the averaged-accuracy of the plane for all the leave-one-out training results. Also, the accuracy of the test itself is the averaged-accuracy of all the leave-one-out test results.

For a wide range of ν parameters, the above process is repeated to find the optimized ν. Training and testing accuracy along the ν parameter is shown in Figure 6.3. The maximal product value of training and testing accuracies along ν is then chosen. At $log2(\nu) = -4$, this value reaches a maximum of 60.8%, with 80.0% and 76% for training and testing accuracies, respectively. It is also noted that the values of the two terms in the Lagrangian function, Eq. 3.3, are close to each other at this ν value, about 6×10^{-2}, respectively. The parameters of the averaged leave-one-out separation plane are listed in Table 6.2, together with the normalization

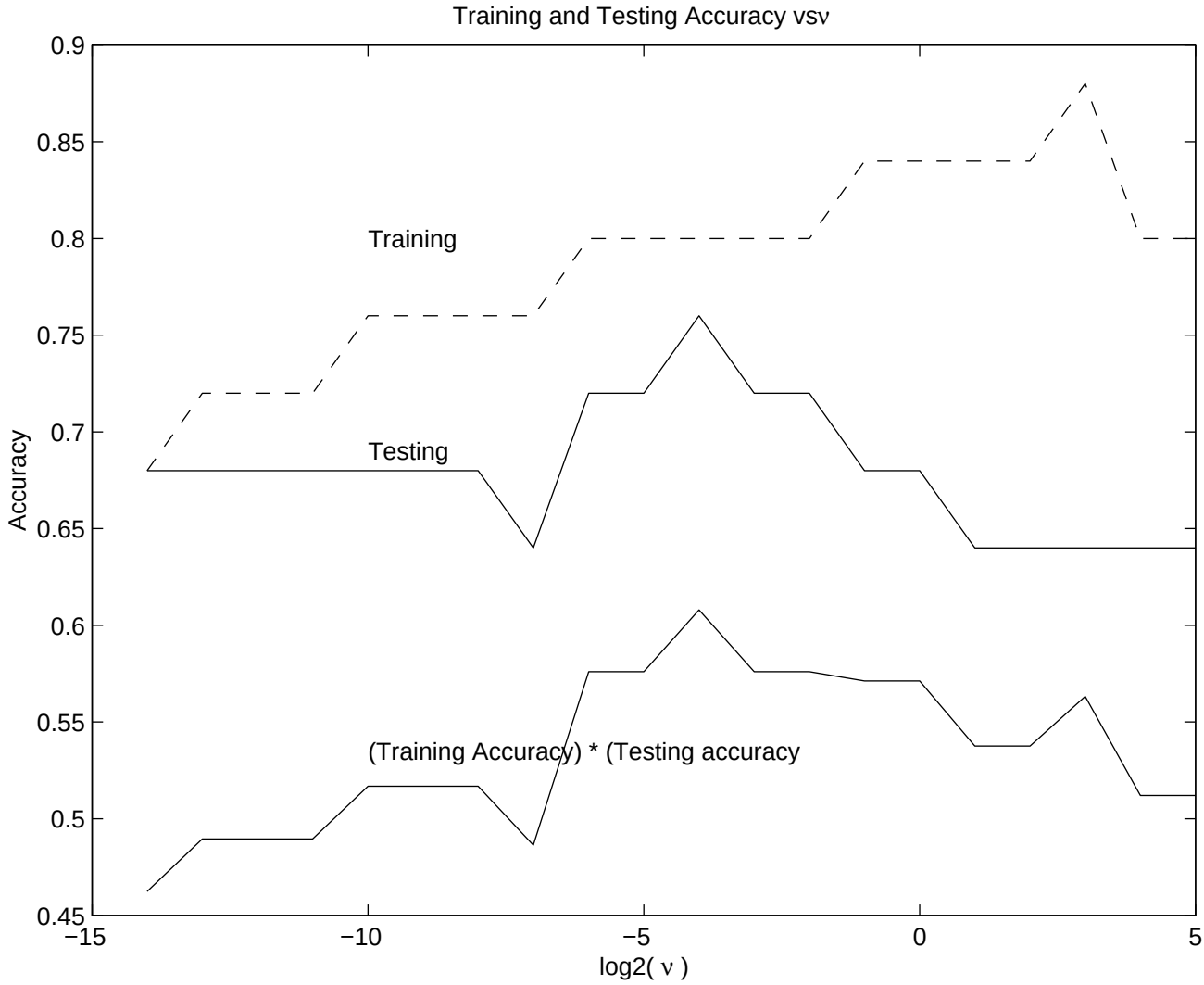

Figure 6.3: Accuracies for training and testing along the selection of the weighting parameter of ν. At $\nu = 1/16$, training and testing accuracy compromise to reach a maximum in terms of their product, shown as the bottom curve.

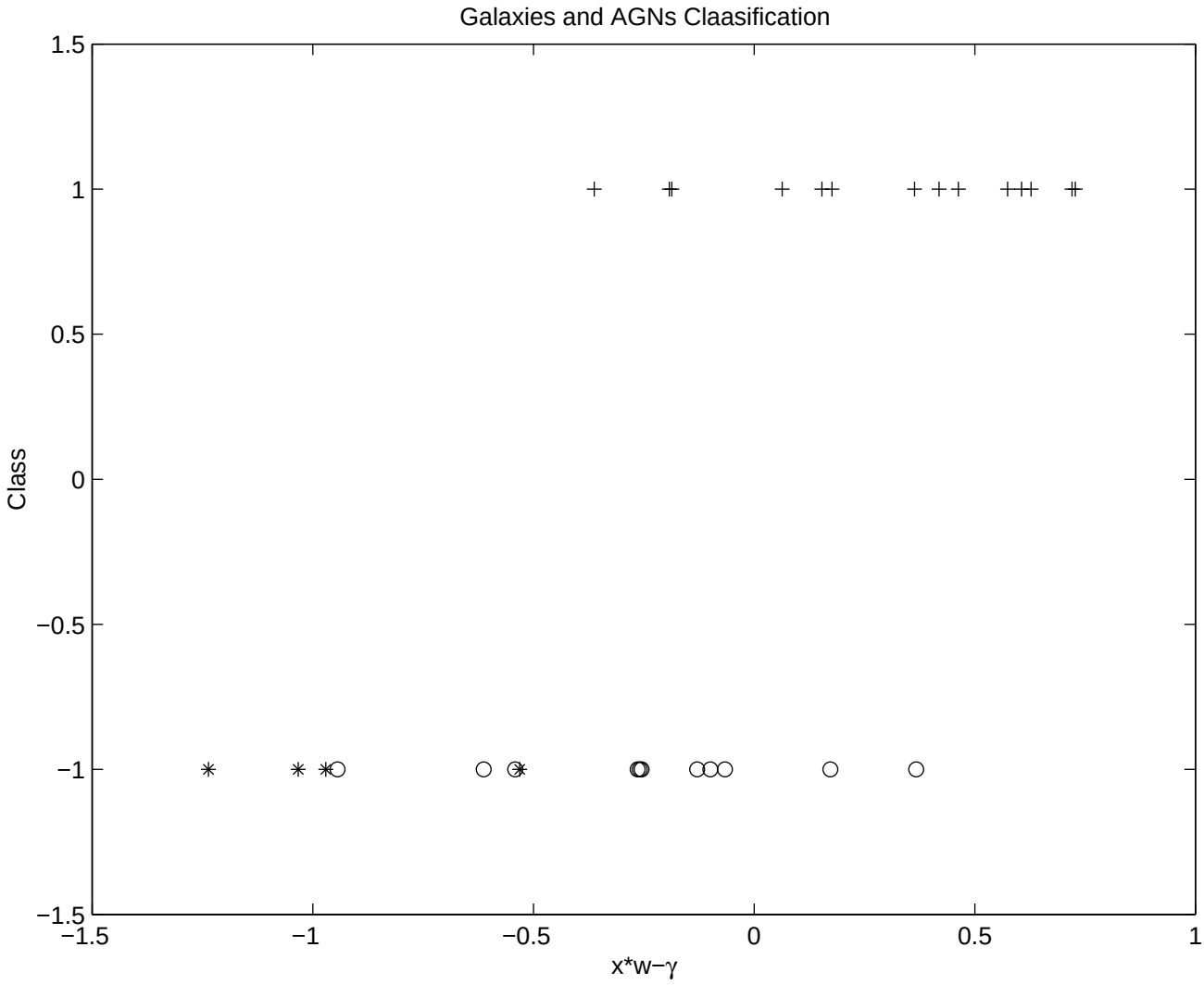

Figure 6.4: Classifi cation of galaxies (o) from AGNs (+). the separation plane is at $f = x * w - \gamma = 0$.

parameters of the components.

The final accuracy percentage for the whole sample, obtained by using the averaged separation plane, is 82.8%. A plot of the classification is shown in Figure 6.4. [1] (Fig. 13), [13] (Fig 8) and [3] show the classification of AGNs and galaxies based roughly on the line of division at $log(Fx/Fr) = -1$, where Fx and Fr are fluxes at the soft X-ray (0.5-2.0 keV) and the R band in optics. Data under this separation regime are plotted in Figure 6.5 for compari-

son. Eight out of 29 objects were misclassified, which corresponds to a 72.4% accuracy. With the ULEDS-SVM separation, as shown in Figures 6.5 and 6.6, there are 5 misclassifications. The accuracy of the ULEDS-SVM classification is 82.8%, better than that obtained from the conventional $log(Fx/Fr) = -1$ separation method.

In order to compare results obtained from ULEDS-SVMs with those from conventional SVMs which do not contain data limits and errors, one may trim the data in one of three possible ways:

1) eliminate the attribute of the HB X-ray flux which contains the greatest number of sources with upper limits, and eliminate source No. 23 which has limits at two optical magnitudes. These compromise the two other trims below;

2) eliminate all data with upper limits, namely, source Nos. 10, 15, 23 and 25;

3) eliminate all attributes with upper limits, namely, I, R, and HB;

The results of the first and the second approaches are shown here for comparison. Approach 3 trims three attributes due to four sources; this approach is expensive and, therefore, not favorable here. In the first approach, there are 28 samples with 8 attributes. A leave-one-out training and testing process was conducted, as described above. This result is also listed at the bottom of Table 6.2. This approach leads to an accuracy of 82.1%, which is about the same as that with all data and attributes included. The margin is $1/norm(w) = 5.15$, also about the same as that of $1/norm(w) = 5.57$ with all data and attributes included. The third approach resulted in an 80.0% accuracy, slightly smaller than the ULEDS-SVM result of 82.8%. It is

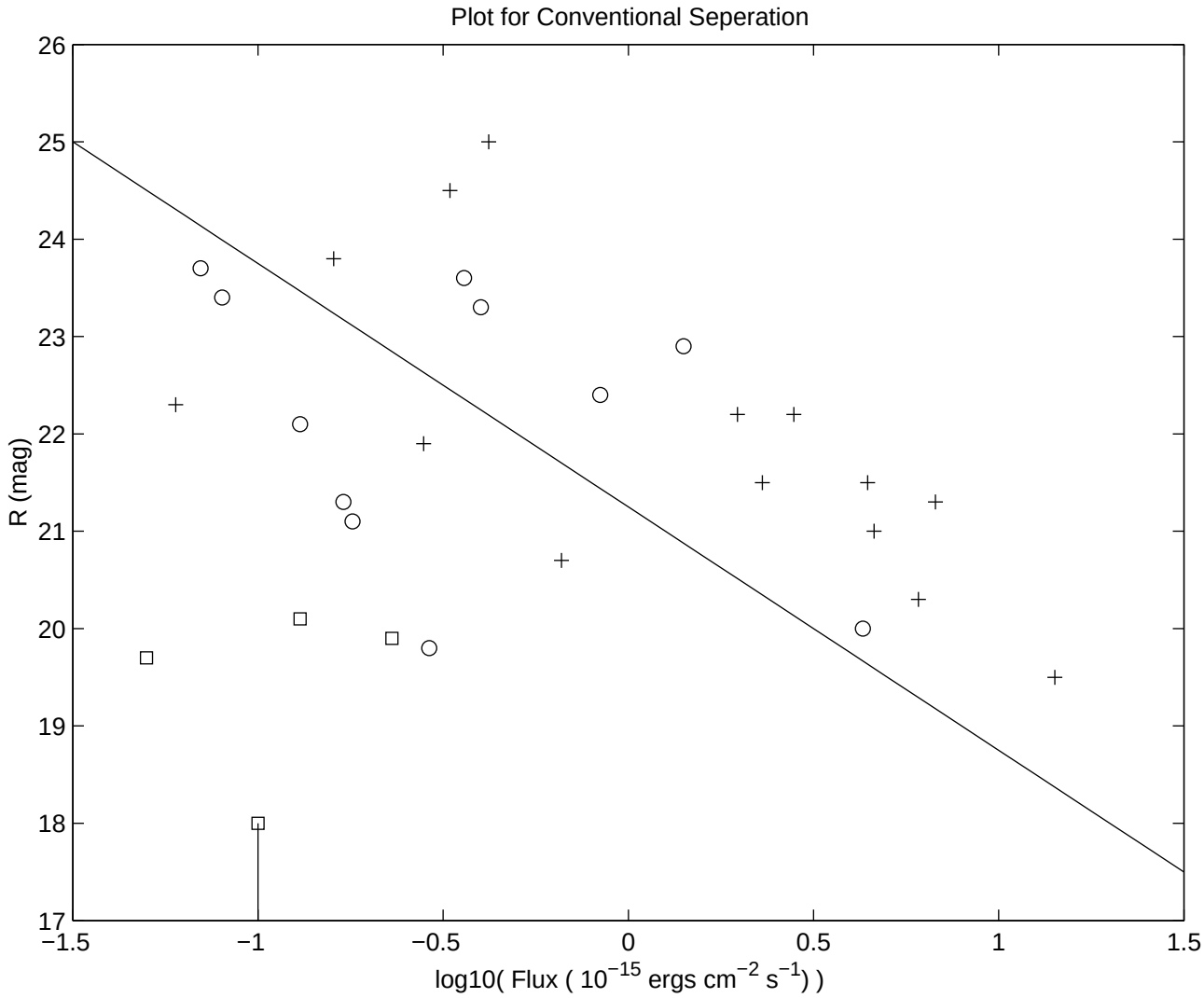

Figure 6.5: Conventional separation of galaxies (o and square sign) from AGNs (+) on R-log(Flux) plot. The separation line is $log(fx/fr) = -1$, where fx/fr is the ratio of the X-ray flux to that at the R band [1] [13] [3]. There are more misclassifi cations here than with the SVM regimes in Fig. 6.6.

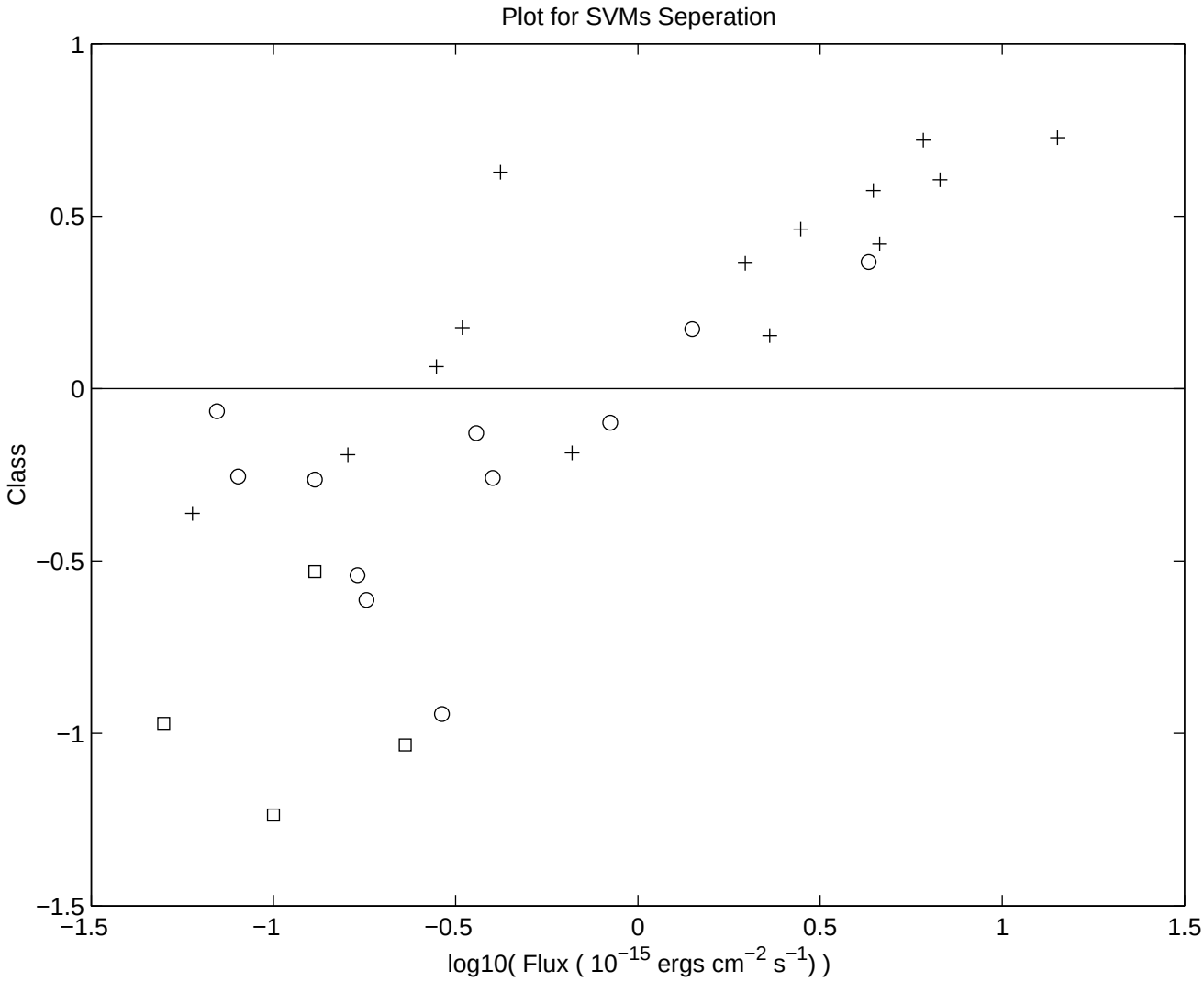

Figure 6.6: SVMs separation of galaxies (o and square sign) from AGNs (+) on Class-log(Flux) plot. The separation line is on the X-axis. There are fewer misclassifi cations here than with the conventional classifi cation regimes in Fig. 6.5.

noted that the HB, SB and FB are not fully independent. The HB may be approximated by subtracting the SB from the FB. This may explain why the difference is not large between the accuracy rates of the trimmed samples and the full sample analysis.

6.4 Prediction

Using the algorithms described in Chapters 3 and 4, the class for extragalactic objects among the 274 sources in the CHDF-N data set (Table 9.1 in Appendix A) was predicted. There are 9 magnitude or flux columns in the Table. The X-ray flux is converted into a $2.5log(Fx)$ scale in order to match up the log scale of the flux at the optical energy band. Table 9.1 also provides the code for upper and lower limits as well as definite values for the corresponding magnitude or flux: 1 for upper limits, -1 lower limits, 0 for definite values. Among these sources are 138 sources with either upper or lower or both kinds of limits among the 9 attributes. This is about half of the total CHDF-N sources compiled here. There are 136 sources that only have definite values.

Predictions were made using the ULEDS-SVM algorithms, implemented with Matlab; the results of the class predictions are given in the C_{pre} column in Table 9.1. Here, 1 is the AGN, -1 is the galaxy, and 9 unpredictable, which means that given the magnitudes and flux limits it is not possible to determine on which side of the separation plane the source point would be. The predictions for the 136 sources with definite values for their classes is straightforward. Among the 138 sources with upper and lower limits, there are 67 unpredictable sources and 71 predictable sources. The total prediction rate for all 274 sources is 75.5%. This is significantly

higher than the 49.6% prediction rate using conventional SVMs.

Note that in Table 9.1 there is no attempt to predict the star class even though there was training for the star class, and it was possible to separate the stars. The reason for this is the limited sample size of stars for training and the uncertainty about their magnitudes. Technically, it is not a significant challenge to separate for the star class. For example, a shorter exposure time would eliminate saturation; moreover, observation from another sky direction that has similar absorption would enlarge the star sample for training. These efforts can help define the separation plane. However, star classification and separation are not the focus of this paper.

Chapter 7

Discussion and Conclusion

7.1 Discussion

Based on the theorems outlined in Chapter 3, the SVMs' treatment in a domain which includes data with upper/lower limits and errors in training can be provided with only one more constraint put on their trainability over and above the ordinary SVM training regime for definite data. This trainability can be tested in a very compact form for limit data and error data. The time complexity of the trainability test is $O(n)$ for an entity that has only limits, and $O(n^2)$ for an entity that has errors or errors mixed with limits, where n is the number of the attributes. The time complexity for the whole training process is $O(\lambda mn)$ and $O(\lambda mn^2)$ for the two cases, respectively, using the algorithms provided in 4.3.1 and implemented in cooperation with LSVM in this case, where λ is the number of iterations for training, and m is the number of entities.

In contrast, the time complexity would be $O(mk^n)$ using the enhanced 1R rule for handling

cases that include data with limits and errors (Chapter 2), where k is the number of elements along one dimension of the attributes. It is obvious that even with this simple 1R, the scalability is challenged to handle the error and limit data, whereas ULEDS-SVMs do not have such a serious problem.

The algorithms provided in the first two sections in Chapter 3 have high portability to assemble conventional SVMs into ULEDS-SVMs in order to support limit and error data. Such portability is an advantage in software engineering. Even in the use of algorithm 4.3.1 to further improve speed, the modification of conventional SVMs is minimized; one must only add the Lagrangian multipliers vector, α, as input and output parameters and skip the initialization step of α within conventional SVMs.

The potential application of ULEDS-SVMs may vary within scientific research and extend to many other fields. One of the issues in astronomy that is relevant to the application of this method concerns the survey study. Most survey studies are based on the source selection of a specific energy band. CHDF-N is such an example where the source completeness is measured in terms of the X-ray source limits. A fundamental question is why would the source selection be based on X-ray or any other specific energy band. A volume limited selection, for example, could possibly lead to a result that is statistically quite different from that of a survey that has as its source selection a specific energy band. For example, the so-called X-ray quiet source would show up and thereby make a difference in the source population. In fact, there are more optical sources than X-ray sources in CHDF directions within the current astronomical facility limits. To conduct a survey such as one that is volume or flux limited, one needs to have a

methodology to handle the mismatch of the sources at different energy bands. That is, a source may show up in some of the energy band observations, but not in others. ULEDS-SVMs may provide one of the tools to handle this classification issue in similar kinds of surveys.

7.2 Further Work

The prospect of generalizing ULEDS-SVMs into a kernel version for nonlinear SVM extension depends on the conditions under which the transformation from attribute space into feature space still maintains the meaning of the upper and lower limits in this space, and probably in others:

1) the upper (lower) limit or error definition is still meaningful in feature space;

2) the intersection definition for data with upper (lower) limits or errors exists in feature space.

One example is logarithmic mapping, which is relatively trivial. In fact, any monotonic mapping can reserve the above two conditions. This space is rescaled with one-to-one dimension mapping. The number of dimensions does not change, and even the dimension itself does not change into any other. Obviously, this is a severe constraint. Finding more general mapping methods which satisfy ULEDS-SVMs may be challenging.

The training algorithms shown in Chapter 4 to treat limits are similar to the perturbation theory in physics which is conditional under some perturbation conditions. Mathematically, this phenomenon is exhibited in the Lagrangian function, 3.3. Its differential deviations introduce no new terms within a reasonable variable range in comparison to its conventional form. That is, it is true that support vectors maintain the multipliers' appearance. However, changing

a component sign of a hypothetical plane normal, if it has a limit, can result in the appearance or disappearance of the corresponding term in Lagrangian. When the evolution of the hyperplane along the course of the iteration step in training is not large, such a term may come and go and can be regarded as perturbation, given that the hypothetical separation plane sweeps gradually in the feature space in the training process. For error data, there could be a more complex process, which is worth probing further.

It has not yet been proven mathematically that ULEDS-SVMs converge within a frame of any SVM in general. However, it can be shown that the implementation of ULEDS-SVMs within the frame of LSVM converge because of the general satisfaction of the $0 < \alpha < \nu/2$ condition for the regression parameter, α, in SVMs. Convergence under more general conditions for other SVMs needs to be studied in further work.

The training algorithms for the limit case in section 4 assume that there is at least one definite value entity in each class. In the case of an empty class set, the initial separation plane must be set in such a way that all data are at one side of the separation plane. Otherwise, the training could lock. One can get around this problem, if there really is an empty class for ordinary data, by some treatment of the initialization handling process. This is not the core part of the algorithm and is not extended here. Besides, for most real world situations none of the sets are empty. Even without such initialization handling, the algorithms can be applied in most of the cases.

Support vectors at the centroid value were chosen if data with errors were trainable. An alternative choice may be the boundary at the end closest to the separation plane. This may be

done by modifying the 4.3.1 algorithm at the corresponding d_ret setting. The current data set is limited to the domain which contains only data with limits.

7.3 Conclusion

The Upper/Lower Limit and Error Data Supported Support Vector Machines, ULEDS-SVMs, were developed in cooperation with data training, testing and prediction by using data with upper or lower limits and errors. The significance of the existence of uncertain data along some of the attribute dimensions is that some of the data may be classifiable and predictable by class, and some may not be classifiable or predictable with respect to the separation plane. The feasibility of classification and prediction was recognized and treated in training, testing and prediction for upper/lower limit cases and for error cases, respectively.

7.3.1 Upper/Lower Limit Cases

- data with upper (lower) limits are not trainable if $w_\mu y \leq 0$ ($w_\mu y \geq 0$). Training can be done by eliminating the untrainable data. Testing using unpredictable data always leads to a true classification result since the separation plane hypothesis does not contradict its class.

- data with upper (lower) limits are trainable if $w_\mu y > 0$ ($w_\mu y < 0$). Training can be done by setting the values of the attributes which have the same limit measurements as the corresponding limit values. Testing can be done the same way as conventional testing of

$(w'x - \gamma)y \geq 0$, but by replacing x^p with its limit value.

- Prediction may have one more category, unpredictable. If there is any intersection between the separation plane and the upper or lower limit of an attribute of an entity, it is not classifiable via the separation plane. Otherwise, the sign of $f = w'x - \gamma$ decides if it is of the "+1" or "-1" class.

7.3.2 Data with Error

The methodology used to classify data with error is a natural extension of that used for data with limit measurements, that is, the simultaneous treatment of the upper and lower boundaries as upper and lower limits.

- The training and prediction feasibility condition is given by $({}_{\mu}\hat{x}'w - \gamma)({}_{\mu}\check{x}'w - \gamma) > 0$ along every attribute dimension. Otherwise, data are untrainable and unpredictable. When the condition is satisfied, both the upper and lower boundaries are on the same side of the separation plane.

- The untrainable data are not included in the sample in training, but they do count towards the number of correct classifications since the hypothesis does not contradict the current classification scheme. Training the trainable data uses the same method as the conventional method by using the centroid value of the attributes of the entity of the data.

- For unpredictable data, testing is set correct. For predictable data, testing is set against $(w'x^p - \gamma)y^p \geq 0$.

- Unpredictable data are not predictable. For predictable data, the data are predicted by the sign of $(w'x - \gamma)$, with "+1" when positive and "-1" when negative.

7.3.3 Application in CHDF-N

The Chandra and Hubble Deep Field North (CHDF-N) database was integrated and consists of 31 identified entities out of 274 total entities. These data sets represent the observations in the sky direction where the largest astronomical facilities allow the longest exposure times in order to study the formation and evolution of galaxies. Some entities have attributes of upper or lower limits because they have reached their detection limits or are saturated in their images. This data set was used as a test bed for ULES-SVMs.

Machine learning via ULES-SVMs was carried out for classification into two kinds of astronomical objects, galaxies and AGNs, by combining their subclassifications; this was done because of the limited number of samples. Even though the extragalactic objects were successfully distinguished from stars, and the power of ULES-SVMs is obvious in eliminating some of the ambiguity, it is not the intention here to emphasize this capability so as not to overread the results of star classification due to the limited statistics available for the star sample.

The identified extragalactic objects were trained and tested using the leave-one-out method. This method obtained an accuracy of 82.8%, which was a significant improvement over the accuracy obtained via the conventional $R - log(Fx)$ plot categorization regime of about 72.4%. Using conventional SVMs, and excluding the upper/lower limit cases by eliminating an attribute and one source from the sample, yields a 82.1% accuracy, obtained from one of the

most optimized approaches; this accuracy rate is about the same as the ULEDS-SVM result. The margin using this approach is 5.15, also about the same as the 5.57 obtained by using ULEDS-SVMs for the whole of the identified data set. Trimming all the limit data results in a 80.0% accuracy. This shows, on the one hand, that the ULEDS-SVMs training provides the best results even though improvement is limited. On the other hand, there may be sufficient margin to warrant further selection of the attributes so that classification is more sensitive to the data and attributes.

Chapter 8

SVMs - a brief history and its updates

Support Vector Machines (SVMs) have recently been developed within the framework of statistical learning theory [25] and have been used successfully in a number of applications, ranging from time series prediction [9] [23] to face recognition to biological data processing for medical diagnosis [11] [17]. Their theoretical foundations and their experimental successes encourage further research on the characteristics of SVMs as well as their further use.

Some articles and books provide excellent reviews and tuition about SVMs, e.g., [5] [7] [24] [8]. In the first part of this section the basic ideas and principles of SVMs are described. Then, in the second part of this section, more recent progress of their implementations is outlined.

8.1 SVMs - as a Statistical Learning Method

8.1.1 Theoretical Framework

Support Vector Machines have been developed and can be incorporated within the framework of Statistical Learning Theory (SLT) [25]. SLT concerns the problems of supervised learning in the following form: here is a set of l training data, $\{(x_i, y_i)\}, i = 1...l$ in $R^{n\times 1}$, sampled according to an unknown probability distribution, $P(x, y)$, and a loss function, $V(y, f(x))$, that measures the error when, for a given x, $f(x)$ is predicted instead of the actual value, y. The problem lies in finding a function, f, that minimizes the expectation of the error on new data; that is, one must find a function, f, that minimizes the expected error:

$$\int V(y, f(x))P(x, y)dxdy \tag{8.1}$$

Because $P(x, y)$ is unknown, one needs to use some form of induction principle to infer from the available m training examples a function that minimizes the expected error. The function uses, as the principle, Empirical Risk Minimization (ERM) over a set of possible functions, which is called hypothesis space, H. This can be written to minimize the empirical error:

$$\frac{1}{l}\sum V(y_i, f(x_i)) \tag{8.2}$$

with f being restricted so as to be in a space of functions, or the hypothesis space, H. Vapnik and Chervonenkis [26] describe the boundary between the expected error and the empirical

error in terms of VC-dimension:

Theorem 4: if V is the VC-dimension of a hypothesis space, H, then with probability, $1-\eta$, the minimum of the expected error that can be achieved with functions from H, say L, and the minimum empirical error, say L_{emp}, satisfy the constraint:

$$||L - L_{emp}|| \leq 4\sqrt{2}\sqrt{\frac{h(log(2l/h)+1) - log(\eta/4)}{l}} \tag{8.3}$$

independent of the distribution of the data, $P(x,y)$.

The VC-dimension for the set of functions, f(x,y), is defined as the maximum number of training points that can be shattered by $f(x,y)$ in H. Here, if a given set of l points can be labeled in all possible 2^l ways in the binary class, and for each labeling a number of the set, $\{f(x,y)\}$, can be found which correctly assigns those labels, one says that the set of points is shattered by that set of functions.

According to the theorem, the higher the VC-dimension, the more complex the hypothesis space and the larger the difference between the expected error and the ERM, as shown in Figure 8.1.

The objective of SLT is to construct the hypothesis spaces, $H_1 \in H_2 \in ... \in H_m$, in order of complexity; i.e., H_{i+1} is more complex than H_i. ERM is performed in each of these spaces, and the choice of the final solution can be made using the aforementioned boundaries. This approach is the so-called Structural Risk Minimization (SRM) [25].

In [27], the VC-dimension was estimated via simulation. By taking into account the probability distribution of $P(x,y)$, which does not appear in Theorem 4, further constraint on the

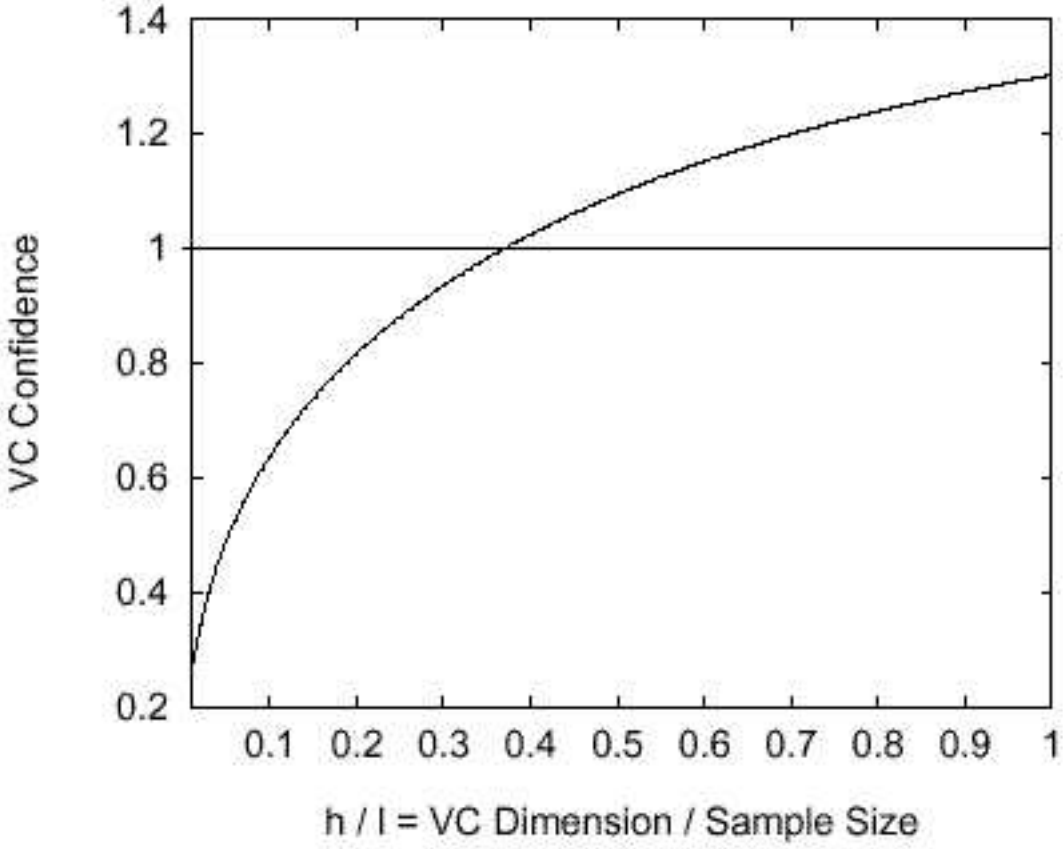

Figure 8.1: VC confidence $||L - L_{emp}||$ vs h/l. Courtesy of [5]. Sample size is 10,000, and η is 0.05, or $1 - \eta = 95\%$. For different η, the trend of the curve is about the same. Sample size is not sensitive as $log(\eta)/l$ is negligible in comparison to the first term of the VC confidence inequality in the theorem.

error estimation may be achieved.

8.1.2 Support Vector Machine

The idea outlined above is adopted in SVM formulation. For SVMs, the objective is to find an optimal hyperplane as the solution to the learning problem. The simplest solution of SVMs is the linear in the attribute space of the data set, x. Here, the hypothesis space is a subset of all the hyperplanes of the form:

$$f(x) = wx + \gamma \tag{8.4}$$

To generalize, SVMs find a hyperplane in a space different from the attribute space. This space is called feature space. It is a hyperplane induced by a kernel, with the definition of the dot product as a subset of Hilbert space. This space can be formally written as

$$\{f : ||f||_k^2 < \infty\} \tag{8.5}$$

where K is the kernel that defines the Reproduction Kernel Hilbert Space (RKHS), and $||f||_k^2$ is the RKHS norm of the function, f. In linear space, $K(x_1, x_2) = x_1 x_2$, and the functions are of the form, $f(x) = wx + \gamma$, and the RKHS norm of these functions is the norm of w, namely $||f_k^2 = ||w||^2$. SVMs consider the subset of this space, namely

$$\{f : ||f||_k^2 \leq A^2\} \tag{8.6}$$

for some constant, A.

Instead of searching many hypothesis spaces one by one using ERM for each choice of A, SVMs search for an A in such a way that the problem changes into a quadratic programming (QP) problem after combining the loss function as a measure for tradeoff.

At this point in the discussion, a distinction is made between the SVM classifier and the SVM regressor. The misclassification error in classification needs to be minimized. So, a loss function of the form sign $(-yf(x))$ should be used. However, by combining the mixed case where two classes are not separable, a soft margin is introduced which can be described by slightly extending the original loss function into the form of $|1 - yf(x)|_+$, where $|a|_+$ is defined as a if $a > 0$ and 0 if $a \leq 0$. The margin has an important geometric significance; for every point, (x, y), its margin is its Euclidean distance from the hyperplane, with a positive sign if the point is correctly classified by the hyperplane, f, and a negative sign otherwise. The margin of the hyperplane, f, is defined as the distance of the closest correctly classified point to the hyperplane, f.

For regression, the loss function used is the so-called epsilon-insensitive loss function, $|y - f(x)|_\epsilon$, which is equal to $|y - f(x)| - \epsilon$ if $|y - f(x)| > \epsilon$, and 0 otherwise. So, the machine learning via SVMs is the minimization problem of

$$min||f||_k^2 + \nu \sum |1 - y_i f(x_i)|_+ \qquad SVM \quad Classification \tag{8.7}$$

$$min||f||_k^2 + \nu \sum |y_i - f(x_i)|_+ \qquad SVM \quad Regression \tag{8.8}$$

where ν is a so-called regularization parameter that controls the tradeoff between the empirical error and the complexity of the hypothesis space that is used.

8.1.3 Lagrangian Approach

The hyperplane in a linear space is

$$f(x, y) = (xw + \gamma) = 0 \tag{8.9}$$

And the margin can be written as the planes of the form:

$$f(x, y) = y(xw + \gamma) - 1 = 0 \tag{8.10}$$

with the "+1" class expected beyond $f(x_i, y_i) = y_i(x_i w + \gamma) - 1 > 0$, and the "-1" class expected beyond $f(x_i, y_i) = y_i(x_i w + \gamma) - 1 < 0$ when it is separable. This can be expressed in term of the Lagrangian function similar to that used in physics.

With the introduction of Lagrangian multipliers, $\alpha_i, i = 1, ..., m$, one has

$$L_P = \frac{1}{2}||w||^2 - \sum \alpha_i y_i (x_i w + \gamma) + \sum \alpha_i \tag{8.11}$$

One must minimize L_P with respect to w, b, and simultaneously require that the derivatives of L_P, with respect to all α, vanish with subjection to the constraints, $\alpha_i \geq 0$. This is a convex quadratic programming problem. The dual problem, the maximization of L_D, also satisfies this

constraint.

Requiring that the gradient of L_P, with respect to w and b, vanish assumes the conditions:

$$w = \sum \alpha_i y_i x_i \tag{8.12}$$

$$\sum \alpha_i y_i = 0 \tag{8.13}$$

Substituting them into Eq. (8.11) gives

$$L_D = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x_i x_j \tag{8.14}$$

Karush-Kuhn-Tucker (KKT) conditions link the optimization problem of the Lagrangian function with the solution of the problem. For the primal problem above, the KKT conditions may be stated ([10]):

$$\frac{\partial}{\partial w_\nu} L_P = w_\nu - \sum \alpha_i y_i x_{i\nu} \tag{8.15}$$

$$\frac{\partial}{\partial \gamma} L_P = - \sum \alpha_i y_i = 0 \tag{8.16}$$

$$y_i(x_i w + \gamma) - 1 \geq 0 \qquad i = 1, ..., m \tag{8.17}$$

$$\alpha_i \geq 0 \qquad \forall i \tag{8.18}$$

$$\alpha_i(y_i(w x_i + \gamma) - 1) = 0 \qquad \forall i \tag{8.19}$$

When there is a mix of the classes, slack variables, ξ_i, are introduced into the function

concerning the margin separation planes (equation 8.10),

$$f(x_i, y) = y_i(x_i w + \gamma) - 1 - \xi_i \tag{8.20}$$

$$\xi_i \geq 0 \qquad \forall i \tag{8.21}$$

for each entity of data. The primal Lagrangian is

$$L_P = \frac{1}{2}||w||^2 + \nu \sum \xi_i - \sum \alpha_i \{y_i(x_i w + \gamma) - 1 + \xi_i\} - \sum \mu_i \xi_i \tag{8.22}$$

where the μ_i are the Lagrange multipliers introduced to enforce positivity of the ξ_i, and ν is a parameter to be chosen by the user to penalize the error in the mixed parts. Its dual form is the same as Eq. 8.14

$$L_D = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x_i, x_j \tag{8.23}$$

except the Lagrangian Multipliers are subject to

$$a \leq \alpha_i \leq \nu \tag{8.24}$$

$$\sum \alpha_i y_i = 0 \tag{8.25}$$

with the boundary of μ rather than 1.

8.1.4 Learning in Nonlinear Space with Kernel

In a generalized nonlinear space, a map from the attribute space into the feature space

$$\phi : R^d \rightarrow H \tag{8.26}$$

exists which maintains the kernel property if the condition of Mercer's theorem is satisfied (see, e.g, [24][10] for details). Using Mercer's theorem, the expensive calculations required to project samples into the feature space can be replaced by a much simpler kernel function, $K(x_1, x_2) = \phi(x_1)\phi(x_2)$, in place of the dot product, $x_1 x_2$, in the Lagrangian dual form in Eq. 8.14. Moreover, one does not need to do the actual mapping, but only refer to the kernel in the feature space.

8.1.5 Multiclasses

The multiclass problem can be solved using two different strategies. One is the so-called one-against-rest [14] method. Each SVM distinguishes one class from all the other classes. The final output is the class that corresponds to the SVM with the highest output value. The other strategy is called the one-against-one [16] method. This method constructs all possible $K(K-1)/2$ binary SVMs for the K-class problem, each of which is used to discriminate two of the K classes only. When a testing pattern is to be classified, it is presented to all the SVMs, each providing a partial answer that concerns the two involved classes.

8.2 SVMs Implementation and Development

8.2.1 Training via Decomposition

There are many implementations for SVMs. An analytical solution can only be possible for a very small number of training data. For training that does not use large amounts of data and attributes, quadratic programming routines can be called on directly to solve the QP problem, derived as Eq. 8.15-8.19. However, the problem quickly exceeds the computer capacity in both time and memory required as the number of data entities and attributes increases if the QP routines are directly called upon. One of the directions of the research in SVM implementation is to deal with the large data set problem. Segmentation is widely used to chuck data into portions in training.

Osuna and Girosi [21] describe a decomposition algorithm to partition the training data into two segments. One is called the working set, B, the other, N. Two propositions hold for the "build down" and "build up" formulation of SVMs:

Proposition 1: moving a variable from B to N leaves the cost function unchanged, and the solution is feasible in the subproblem.

Proposition 2: moving the variable that violates the optimal conditions from N to B gives a strict improvement in the cost function when the subproblem is re-optimized.

The algorithm based on these two propositions is:

1. Arbitrarily choose $|B|$ points from the data set.

2. Solve the subproblem defined by the variables in B.

3. While some $j \in N$ exists that satisfies the support vector condition, move it to B and solve the new problem in B.

The propositions above can be seen as the basis for the category of decomposition algorithms that deal with large sets of data in SVMs. These products include:

BSVM which provides SVMs with the solution for large classification and regression problems [[14]]. Its implementation borrows the structure of LIBSVM (see below). For the boundary-constrained formulation for classification and regression, BSVM uses a decomposition method. The use of a special implementation of the optimization solver, TRON, allows BSVM to identify bounded variables.

LIBSVM [6] is a library containing software for support vector classification (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multiclass classification. The basic algorithm is a simplification of both the Sequential Minimal Optimization (SMO) by Platt and SVM^{Light} by Joachims. It is also a simplification of modification 2 of the SMO by Keerthi et al. These are starting practice tools to help ease the complexity of SVM applications in general.

SVM^{Light} [15] is another implementation based on the decomposition algorithms. It is a package that is able to handle large data sets. It applies some sophisticated methods to obtain the set of variables that is moved up to or down from the B set, accompanied by expanding or shrinking the cache size, and making efficient housekeeping of the decomposition, gradient and termination.

The SMO [[**?**]] can be seen as the extreme case of decomposition where the B set contains

only two variables. Then, an analytical solution can be obtained directly, freeing the innermost iteration from the code as opposed to the ordinary case for B set optimization problem solving. This saves a calling procedure for the QP problem solver at the innermost nest level.

8.2.2 LSVM

The Lagrangian Support Vector Machine (LSVM) [[20]] belongs to another kind of algorithm, other than the decomposition algorithm. Because of its simplicity and its use as the basis of the ULEDS-SVMs application in CHDF-N data in this work, it is described in a little more detail here.

Some modifications of the standard Lagrangian Function (Eq. 8.22 and 8.23) are necessary: one is to add a term so that the distance of the plane is measured in $(w, \gamma) \in R^{n+1}$ space; another is that the term of the slack variables appears quadratically rather than in linear form, as used before. In fact, it is as reasonable to use the quadratic term as it is to use a linear term, since it is used to measure the penalty for the misclassification. Both can effectively achieve this penalty purpose. The new function appears as in Eq. 8.27.

$$\min_{(w,\gamma,y)\in R^{n+1+m}} \nu\frac{y'y}{2} + \frac{1}{2}(w'w + \gamma^2) \quad \text{s.t. } D(Aw - e\gamma) + y \geq e. \tag{8.27}$$

As a result, its dual form is a positive definite:

$$\min_{0\leq u\in R^m} \frac{1}{2}u'(\frac{I}{\nu} + D(AA' + ee')D)u - e'u. \tag{8.28}$$

By defining the notations of

$$H = D[A \quad -e], \qquad Q = \frac{I}{\nu} + HH'. \tag{8.29}$$

Equation 8.28 takes the form of

$$\min_{0 \le u \in R^m} f(u) := \frac{1}{2}u'Qu - e'u. \tag{8.30}$$

This reaches a concise KKT condition of

$$0 \le u \ \perp \ Qu - e \ge 0. \tag{8.31}$$

These optimal conditions lead to the following form to be used as the basis for the iteration formula of the LSVM algorithm:

$$u^{i+1} = Q^{-1}(e + ((Qu^i - e) - \alpha u^i)_+), \ i = 0, 1, \ldots, \tag{8.32}$$

which converges from any starting point under the condition of

$$0 < \alpha < \frac{2}{\nu}. \tag{8.33}$$

Here, i represents the number of iterations. The algorithm iterates using equation 8.32 directly, and obtains the result of the Lagrangian Multipliers when converged under the termination

condition. The results regarding the separation plane are:

$$w = A'Du, \;\; y = \frac{u}{\nu}, \;\; \gamma = -e'Du. \tag{8.34}$$

8.2.3 SSVM and RSVM

Smoothing is a kind of treatment to overcome overfit. Overfit problems may occur when there are more parameters than needed to describe a feature. Optimization in a feature space with a large number of parameters for the support vectors is likely to exhibit overfit. In [18]'s Smooth Support Vector Machine (SSVM), the Lagrangian Function is adopted in the same way as it is used in Eq 8.27 for LSVM. Moreover, the slack variable, ξ, is replaced directly by its solution, $\xi = (e - D(Aw - e\gamma))_+$. Then, the integral of the sigmoid function, $\frac{1}{1+\epsilon^{-\alpha x}}$:

$$p(x, \alpha) = x + \frac{1}{\alpha} log(1 + \epsilon^{-\alpha x}), \alpha > 0 \tag{8.35}$$

is used as an approximation to smooth the slack variables in Eq. 8.27. The resultant SSVM takes the form of:

$$\min_{(w,\gamma)\in R^{n+1}} L_\alpha(w, \gamma) : \min_{(w,\gamma)\in R^{n+1}} \frac{\nu}{2} ||p(e - D(Aw - e\gamma), \alpha)||_2^2 + \frac{1}{2}(w'w + \gamma^2). \tag{8.36}$$

The algorithm of SSVM works directly on the Lagrangian Function (eq. 8.36) Two techniques are used: a) the Newton Direction, d^μ, is used to determine the steepest gradient direction of

the Lagrangian Function,

$$\nabla^2 L_\alpha(w^\mu, \gamma^\mu)d^\mu = -\nabla L_\alpha(w^\mu, \gamma^\mu)' \tag{8.37}$$

b) Armijo Stepsize, $\lambda_\mu \in R$, is used so that

$$(w^{\mu+1}, \gamma^{\mu+1}) = (w^\mu, \gamma^\mu) + \lambda_\mu d^\mu \tag{8.38}$$

where $\lambda_\mu = max1, 1/2, 1/4, ...$ such that

$$L_\alpha(w^\mu, \gamma^\mu) - L_\alpha((w^\mu, \gamma^\mu) + \lambda_\mu d^\mu) \geq -\delta\lambda_\mu \nabla L_\alpha(w^\mu, \gamma^\mu)d^\mu \tag{8.39}$$

where $\delta \in (0, 1/2)$.

Using a smoothed SVM technique such as SSVM [18], [19] proposed a reduced SVM, RSVM, to handle very large data sets. The kernel, $K(A, A)$, when replacing the product in linear form, may be replaced by $K(A, \bar{A})$, where $\bar{A}$ is a subset, usually 1%-10% in the original events. Even though half of the kernel is so reduced, the interaction of the two parts still contains information, hopefully enough to construct the separation hyperplane. The smooth SVM, e.g., SSVM, is used in training the set for the reduced kernel. According to [19], in testing several data sets, RSVM does not have a worse performance, and has mostly a better performance, than SMO and PCGC, a Gaussian Kernel SVM, in both time and correctness. It may appear surprising that reducing the data set would produce an even better correctness

result than not reducing it. However, this may be understandable in light of the utilization of the smoothing technique which overcomes overfit. At the same time, the so-called reduced SVM is, in fact, only a reduction of part of the kernel. Every entity of the data set is utilized through the first half of the kernel, at least. In this sense, there may not be much information that is lost.

Chapter 9

Appendix A - CHDF-D Dataset and Class Prediction

The Table in this Appendix shows the Chandra and Hubble Deep Field North (CHDF-N) dataset and its class prediction. In the attribute columns, the code for upper limits, lower limits or definite value is also given under the slash "/". The columns have the same meaning as explained for Table 9.1 wherever common columns are the same (see text). The first column shows the sequence number, and the second column shows the source number from [4]. The last column, C_{pre}, shows the classes predicted using the training results in Table 6.2: 1 for AGN, -1 for galaxy, 9 for unpredictable. (See details in chapter 6 for training, testing and prediction.) Among the 29 extragalactic sources with known identifications, there are 5 misclassifications resultant from the training. The Table shown here contains part of a total of 274 sources. There are 138 sources with limit measurements; 71 sources are predictable, and

67 sources are unpredictable. It shall be noted that there is no effort to pre-separate stars from others in this Table.

The whole Table can be accessible on line (http://www4.ncsu.edu:8030/˜xsun4).

Seq	No.	HK'	z'	I	R	V	B	FB	SB	HB	C
1	20	22.6/-1	26.3/0	26.4/0	29.2/-1	28.5/-1	29.0/-1	-0.1765/0	-1.3821/0	-0.1512/1	9
2	21	20.9/0	24.7/0	24.9/0	25.5/0	26.1/0	26.5/0	-0.7745/0	-2.3020/0	-0.5190/1	9
3	22	22.6/-1	25.2/0	25.1/0	28.0/0	28.5/0	28.5/0	1.3786/0	-0.6893/0	1.1371/0	1
4	23	22.6/-1	27.0/-1	27.6/-1	29.2/-1	28.5/-1	29.0/-1	0.2070/0	-0.9419/0	-0.2980/0	9
5	24	21.6/0	24.6/0	24.7/0	25.5/0	25.7/0	26.0/0	-0.2023/0	-1.8031/0	-0.3873/1	9
6	25	19.8/0	22.6/0	22.5/0	23.2/0	23.7/0	23.9/0	1.1672/0	0.2423/0	0.4617/0	1
7	27	20.1/0	21.9/0	21.7/0	22.9/0	23.4/0	24.2/0	0.4966/0	-1.9897/0	0.3807/0	1
8	28	19.3/0	22.8/0	22.6/0	23.1/0	23.4/0	24.0/0	0.1980/0	-1.9897/0	0.0215/0	-1
9	29	22.6/-1	24.3/0	24.4/0	25.4/0	25.6/0	26.3/0	2.8822/0	1.0744/0	2.6192/0	1
10	30	18.5/0	22.5/0	22.2/0	22.7/0	23.7/0	24.8/0	0.9137/0	-1.5495/0	0.8561/0	-1
11	31	22.6/-1	25.4/0	27.3/0	26.7/0	28.5/-1	27.7/0	3.1860/0	1.8229/0	2.7739/0	9
12	32	18.7/0	21.2/0	21.0/0	21.3/0	21.8/0	22.0/0	3.0568/0	2.0205/0	2.5457/0	1
13	35	20.9/0	22.8/0	22.9/0	24.2/0	26.1/0	26.3/0	1.0744/0	-2.1347/0	1.0248/0	1
14	36	16.7/0	18.0/1	18.0/1	20.0/0	21.0/0	22.5/0	-0.9949/0	-1.6945/0	-0.4348/1	-1
15	37	17.7/0	19.5/0	19.1/0	19.6/0	20.1/0	20.9/0	-0.8670/0	-1.7474/0	-0.8670/1	-1
16	38	18.3/0	20.6/0	20.5/0	21.4/0	22.5/0	23.9/0	2.8290/0	1.5293/0	2.4636/0	1
17	39	18.8/0	22.0/0	22.1/0	23.7/0	24.7/0	25.9/0	2.9539/0	-1.6439/0	2.9716/0	1
18	40	16.2/0	18.1/0	18.0/1	18.8/0	19.7/0	21.2/0	-0.1024/0	-0.6690/0	-0.8670/1	-1
19	41	22.1/0	23.7/0	23.3/0	23.8/0	24.0/0	24.3/0	0.1517/0	-1.6439/0	-0.0443/0	1
20	42	20.7/0	22.4/0	21.9/0	22.2/0	22.4/0	23.2/0	0.2248/0	-0.9163/0	-0.2559/1	9
21	43	20.2/0	25.1/0	24.7/0	25.3/0	25.3/0	25.7/0	-0.5914/0	-1.4216/0	-0.9419/1	9
22	45	18.1/0	20.4/0	20.3/0	21.2/0	22.2/0	23.0/0	-1.0505/0	-2.6144/0	-0.6295/1	-1
23	46	18.3/0	20.2/0	19.9/0	20.4/0	20.7/0	21.3/0	-1.2716/1	-2.8873/0	-0.8670/1	-1
24	47	17.0/0	19.1/0	19.0/0	19.6/0	20.2/0	21.2/0	-1.1092/0	-1.9897/0	-0.9419/1	-1
25	48	20.5/0	22.5/0	22.6/0	23.7/0	24.3/0	24.6/0	0.6199/0	-2.1347/0	0.5697/0	1
26	50	22.6/-1	23.2/0	23.0/0	23.5/0	23.8/0	24.1/0	-0.3719/0	-1.5957/0	-0.8670/0	1
27	52	17.6/0	19.2/0	19.8/0	20.6/0	21.7/0	22.4/0	-1.3440/0	-2.2151/0	-0.9163/1	-1
28	55	18.9/0	21.7/0	21.6/0	22.7/0	23.4/0	24.2/0	0.6969/0	-1.9239/0	0.6561/0	-1
29	56	19.2/0	22.5/0	22.7/0	23.1/0	23.9/0	23.8/0	0.4402/0	-0.8198/0	0.0000/0	1
30	57	21.2/0	26.0/0	25.2/0	26.6/0	26.6/0	29.0/-1	1.3445/0	-0.1024/0	1.0824/0	9

Table 9.1: Chandra and Hubble Deep Field North (CHDF-N) dataset and its class prediction. The column has the same meaning as explained for Table 5.1 for the common columns (see text). In each attribute column, the code for the limit is also given: 1 for upper, -1 for lower limit and 0 for definite value. The last column, C_{pre}, shows the classes predicted using the training results in Table 6.2 1 for AGN; -1 for galaxy; 9 for unpredictable target. The full Table is accessible at http://www4.ncsu.edu:8030/~xsun4

Chapter 10

Appendix B - Matlab code for ULEDS-SVMs

A Matlab code is provided in this Appendix as a reference. Matlab has a convenient matrix manipulation functionality which helps reduce the length of the code. Otherwise, the operation would be the same as coding via another language. One function from this implementation was called upon for the Lagrangian Support Vector Machines, developed by Mangasarian and Musicant [20], which can be accessible via http://www.cs.wisc.edu/dmi/lsvm/ webpage. This conventional SVM can be replaced by any other SVM without losing its generality.

```
function [count,OutC,w_new,gamma_new] = uleds_svms(D, U, C, w, gamma, Op,Tcond);
%
% ULEDS-SVMs: a SVM for data training, testing and prediction on the
%             data domain which includes upper/lower limits and errors
%
% Author: Xuejun Sun
% Date:   June 25, 2003 - initial coding
%
% D R^{m x n x 3} event multiple-matrix for data containing limit and error
% U R^{m x n}     limit matrix for upper/lower/error/definite with 1/-1/0/2
% C R^m           column vector for the class of the m events
% w, gamma        separation plane, w - column (n row) vector, gamma - scalar
% Op              operation code for training, testing and predicting of 1/2/3
% Tcond           not really used at this moment
%
%-------------------------------------------------------------------------
  [m,n,l] = size(D);  % get the number of items and attributes, scalars
  d_ret = ones(n,1);  % point attributes at return of subroutine, col vector
  count = 0;          % counting for test correct number, scalar
  OutC = zeros(m,1);  % output class, column vector

  % branching according to Op - training 1, testing 2 and predicting 3
  if Op == 1          % training

    W = zeros(m,n);
    Wclass = zeros(m,1);

    % initialization of
    d = zeros(n,3); % attributes with lower, centroid and upper values
    u = zeros(1,n); % upper/lower limits and error code
    c = 0;          % class
    iterT = 0;      % 1 true, 0 false
    iterWhile = 0;

    % lsvm relevant parameters
    nu = 1/m;
    alpha = 1.9/nu;
    tol = 1.e-10;
    maxIter = 100;
    perturb = 0;
    normalize = 0;

    count_old = 0;
    W_old = zeros(1,n);
    while iterT == 0   % main loop for trainable data selection and training

      % prepare the set by rejecting untrainable items in training
      count = 0;
      for i = 1:m      % filtering through the dataset
        % extract one item from the dataset
        d = D(i,:,:);
        u = U(i,:);
        c = C(i,1);

        % check out the trainability
        [TTP, d_ret] = trainPredictability_ULED(d, u, c, w, gamma, Op);
        if TTP == 1
          W(i,:) = d_ret;
          Wclass(i,1) = c;
          count = count + 1;
        end;
      end;  % end for loop

      % purify the set for trainable data
      W1 = ones(count,n);
      W1 = W(1:count,:);
      Wclass1 = ones(count);
      Wclass1 = Wclass(1:count,1);

      % invoke LSVM of  Mangasarian & Musicant
      [iterLSVM, optCond, time, w_new, gamma_new] = lsvm(W1,Wclass1,...
          nu,tol,maxIter,alpha,perturb,normalize);

      % setup termination condition
      if count_old == count
        if W_old == W1
          iterT = 1;    % terminate
        end;
      else              % continue
        count_old = count;
        W_old = W1;
      end;

    end;  % end while loop
    count = -1;
    return;
```

```
  elseif Op == 2      % testing
    % test classification item by item via for loop
    for i = 1:m
      % extract one item from the dataset
      d = P(i,:,:);
      u = U(i,:);
      c = C(i,1);

      % check out the test/predictability
      [TTP, d_ret] = trainPredictability_ULED(d, u, c, w, gamma, Op);
      if TTP == 1   % predictable/testable
        if ( c * fplane(d_ret, w, gamma)) >= 0    % right classification
          count = count + 1;
          OutC(i,1) = 1;
        else                                      % wrong classification
          OutC(i,1) = -1;
        end;
      else          % unpredictable item is regarded as right in testing
        count = count + 1;
        OutC(i,1) = 1;
      end;
    end;            % end for loop
    return;

  elseif Op == 3      % predicting
    % predict the class item by item via for loop
    for i = 1:m
      % extract one item from the dataset
      d = P(i,:,:);
      u = U(i,:);
      c = C(i,1);

      % check out the predictability
      [TTP, d_ret] = trainPredictability_ULED(d, u, c, w, gamma, Op);
      if TTP == 1
        % predictable class item
        if fplane(d_ret, w, gamma) >= 0
          OutC(i,1) = 1;
        else
          OutC(i,1) = -1;
        end;
      else
        % unpredictable item
        OutC(i,1) = 9;
      end;
    end;            % end for loop
    count = -3;
    return;
end;

% function about trainability, testability and predictability
function [TP, d_ret] = trainPredictability_ULED(d, u, c, w, gamma, Op);

% d - R^{n x 3} matrix for one item attributes, but with 3 frames for boundaries
% u - R^n       vector for the upper/lower-limit/error/definite code 1/-1/0/2
% c - R^1       class 1/-1
% w, gamma      separation plane as in main program
% Op            operation code as in main program
%
% Return:
% TP            Training-Predicting feasibility: 1 true, 0 false
% d_ret         vector about the attribute values to be used if feasible
%
  n = size(u,1);
  d_ret = d(:,:, 2);        % store the centroid attribute values in d_ret

  % feasibility test about training, testing and predicting through attributes
  for mu = 1:n
    if u(1,mu) == 0
      % case of error attribute
      pup  = p(mu, 3);
      plow = p(mu, 1);
      if intersectCompErr(mu, pup, plow, w, gamma,d_ret) == 1
        TP = 0;
        return;
      end;
    elseif abs(u(1,mu)) == 1
      % case of upper or lower limit
      if Op == 1                % training
        if trainabilityCompLmt(u(1,mu), c, w(mu,1)) == 0
          TP = 0;
          return;
        end;
      elseif ( Op == 2 | Op == 3 )  % testing or predicting
        if intersectCompLmt(mu, u(1,mu), w, gamma, d_ret) == 1
          TP = 0;
          return;
```

```
        end;
      end;
    end;     % end outmost if block
  end;       % end for loop
  TP = 1;
  return;

% function about the intersection test for a component for data with error
function interSErr = intersectCompErr(mu, pup, plow, w, gamma, d_ret);
%
% mu         component
% pup        upper bound of mu component
% plow       lower bound of mu component
% w,gamma    separation plane as described in the main program
% d_ret      vector for attributes to be used in the test
%
% Return
% interSErr  integer value 1 true, 0 false about the intersection
%
  xup = d_ret;
  xup(mu,1) = pup;
  xlow = d_ret;
  xlow(mu,1) = plow;
  if ( fplane(xup, w, gamma) *  fplane(xlow, w, gamma) ) <= 0
    interSErr = 1;    % intersect
  else
    interSErr = 0;    % not intersect at this component
  end;
  return;

% function about the intersection test for a component for data with limit
function interSLmt = intersectCompLmt(mu, u_mu, w, gamma, d_ret)
% mu         component
% u_mu       value for the upper/lower limit code at mu'th component, 1/-1
% w,gamma    separation plane as described in the main program
% d_ret      vector for attributes to be used in the test
%
% Return
% interSLmt  integer value 1 true, 0 false about the intersection
%
  f = fplane(d_ret, w, gamma)
  if f == 0          % one side ending at the separation plane
    interSLmt = 1;
  elseif (f * w(mu,1) * u_mu) > 0
    interSLmt = 1;   % intersection
  else
    interSLmt = 0;     % no intersection
  end;
  return;

% function about the trainability test for a component for data with limit
function trainabilityCompLmt = trainabilityCompLmt(u_mu, c, w_mu);
% u_mu       value for the upper/lower limit code at mu'th component, 1/-1
% c          class of the item to be tested
% w_mu       mu'th component of the separation plane normal, w.
%
% Return
% trainabilityCompLmt 1 true, 0 false
%
  if w_mu * u_mu * c <= 0
    trainPredLmt = 0;
  else
    trainPredLmt = 1;
  end;
  return;

% function about the separation plane function
function fplane = fplane(d, w, gamma)
% d          vector for attributes of one point in space R^n
% w,gamma    separation plane
% Return
% fplane     the function value
%
  fplane = d' * w - gamma;
  return;
```

Bibliography

[1] Alexander, D.M., Bauer, F.E., Brandt, W.N. et al., "The Chandra Deep Field-North Survey. XIII. 2 Ms Point-Source Catalogs", to appear in ApJ, August, 2003

[2] Barger, A.J., Cowie, L.L., Brandt, W.N., et al, "X-Ray, Optical, and Infrared Imaging and Spectral Properties of the 1 Ms Chandra Deep Field North Sources", ApJ, 2002, **124**, 1839

[3] Barger, A.J., et al, "Optical and Infrared Properties of the 2 Ms Chandra Deep Field-North X-ray Sources", to appear in ApJ, August, 2003

[4] Brandt, W.N., Alexander, D.M., Horschemeier, A.E., et al., "The Chandra Deep Field North Survey. V. 1 Ms Source Catalogs" AJ, 2001, **122**, 2810

[5] Burges, C.J.C., "A Tutorial on Support Vector Machines for Pattern Recognition" Data Mining and Knowledge Discovery, 1998, **2**, 121

[6] Chang, C.-C. and Lin, C.-J., "a library for support vector machines", http://www.kernel-machines.org/ software linkage

[7] Cristianini, N. and Shawe-Taylor, J., "An Introduction to Support Vector Machines" Cambridge University Press, Cambridge, UK, 2000

[8] Evgeniou, T. and Pontil, M., "Support Vector Machines: Theory and Applications", eds: Paliouras, G. and Karkaletsis, V., ACAI '99, LNAI 2049, pp 249-257, 2001

[9] Fernández, R., "Predicting Time Series with a Local Support Vector Regression Machine", in ACAI '99, 1999

[10] Fletcher, R., "Practical Methods of Optimization", John Wiley and Sons, Inc, 2nd edition, 1987

[11] Göktürk, S.B., Tomasi, C. et al., "A Statistical 3-D Pattern Processing Method for Computer-Aided Detection of Polyps in CT Colonography", IEEE Transactions on Medical Imaging, 2001, **20**, No. 12, 1251

[12] Holte, R.C., "Very Simple Classification Rules Perform Well", Machine Leaning, 1993, 11:63-91

[13] Hornschemeier, A.E., Brandt, W.N., Garmire, G.P., et al., "The Chandra Deep Survey of the Hubble Deep Field-North Area. II. Results from the Caltech Faint Field Galaxy Redshift Survey Area", ApJ, 2001, **554**, 742

[14] Hus, C.-W. and Lin, C.-J., "A Comparison of Methods for Multi-class Support Vector Machines", IEEE Transactions on Neural Networks, vol. 13, 415-425, 2002

[15] Joachims., T., "Making large-scale support vector machine learning practical", in Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1999

[16] Krebel, U., "Pairwise classification and Support Vector Machines", in Adv in Kernel Methods - Support Vector Learning", pp 255-268, Cambridge, MA, 1999, MIT Press.

[17] Land, W.H.Jr, et al., "Performance Tradeoff Between Evolutionary Computation (EC)/Adaptive Boosting (AB) Hybrid and Support Vector Machine Breast Cancer Classification Paradigms", Proceedings of the 2002 Congress on Evolutionary Computation, CEC2002

[18] Lee, Y.,-Y., and Mangasarian, "SSVM: A Smooth Support Vector Machine For Classification", Data Mining Institute Technical Report 99-03, September 1999

[19] Lee, Y.,-Y., and Mangasarian, "RSVM: Reduced Support Vector Machines", in Proceedings of the SIAM International Conference on Data Mining, Chicago, April 5-7, 2001, SIAM, Philadelphia, ISBN 0-89871-495-8

[20] Mangasarian, O.L. & Musicant, D.R., "Lagrangian Support Vector Machines", Journal of Machine Learning Research, 2001, **1**, 161

[21] Osuna, E., Girosi, F., "An Improved Training Algorithm for Support Vector Machines" in Proc. of IEEE NNSP'97, Amelia Island, FL, 24–26 Sep., 1997

[22] Platt, J., "Sequential minimal optimization: A fast algorithm for training support vector machines", in Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1999

[23] Quiñonero-Candela, J. and Hansen, L.K., "Time Series Prediction Based On the Relevance Vector Machine With Adaptive Kernels", in "International Conference on Acoustics, Speech, and Signal Processing", 2002, pp 985-988.

[24] Schölkopf, B. and Smola, A, "Learning with Kernels", MIT Press, Cambridge, MA, 2002

[25] Vapnik, V.N., The Nature of Statistical Learning Theory, Springer, New York, 1995

[26] Vapnik, V.N. and Chervonenkis, A., "On the Uniform convergence of relative frequencies of events to their probabilities", in "Theoretical Probability and its applications" 1971, **17** (2) 264-280

[27] Vayatis, N. and Azencott, R., "How to Estimate the Vapnik-Chervonenkis Dimension of Support Vector Machines Through Simulations?", in "Support Vector Machines: Theory and Applications", eds: Paliouras, G. and Karkaletsis, V., ACAI '99, LNAI 2049, pp, 2001