LAND COVER CLASSIFICATION USING LINEAR SUPPORT VECTOR MACHINES

by

Mohammad Danish Shakeel

Submitted in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in the

Department of Mathematics and Statistics
College of Science, Technology, Engineering and Mathematics

YOUNGSTOWN STATE UNIVERSITY
December, 2008

Land Cover Classification Using Linear Support Vector Machines

Mohammad Danish Shakeel

Signature:

Mohammad Danish Shakeel, Student Date

Approvals:

Alina Lazar, Dissertation Advisor Date

Jamal Tartir, Committee Member Date

Neil Flowers, Committee Member Date

Peter J. Kasvinsky, Dean of School of Graduate Studies & Research Date

ABSTRACT

GIS has been an effective tool in identifying and recognizing urban patterns. Various techniques like Support Vector Machines, artificial neural networks have been used with GIS to classify the patterns for urban analysis. Liblinear has emerged as another effective tool which produces results in much lesser time without compromising the accuracy. In this thesis the datasets used were extracted using GIS. The datasets were from the Ohio state counties namely the Delaware, Holmes, Mahoning and Medina counties. Each had over a million records and contained seven independent variables related to urban development and a class label which denotes the urban areas versus the rural areas. Using Liblinear, Libsvm, Rapid Miner and Weka some experiments were carried out over smaller datasets and the results have been shown. It can be seen that Liblinear is as effective as Libsvm while the latter takes much longer time for producing the results. The results can help indentify geographical patterns related to urban land use.

ACKNOWLEDGEMENTS

I would like to thank Dr. Alina Lazar for the many opportunities of doing research with her, especially on the topic of Support Vector Machines. This research allows me to pursue my academic goals to the fullest. I would like to thank Dr. Jamal Tartir and Dr. Neil Flowers for being the committee members of my thesis and helping me out on too many occasions to count at YSU. Also, I would like to thank Dr. Lazar for the academic help and moral support provided in and out of the class room.

TABLE OF CONTENTS

LIST OF TABLES

1 Introduction

Accurate and current urban maps can be the most valuable tools for study related to census, economics, culture and public safety. Additional information can also be extracted from urban maps by effectively combining Support Vector Machines (SVM) and Geographic Information Systems (GIS). This information is of extreme importance for analysis of patterns of urban land use [25].

It has been demonstrated [14] that SVM algorithms combined with feature selection procedures provide an efficient classification and prediction method for large real datasets from the US census. Early applications of SVM approaches to geographic data included modeling of land cover classifications [12], which did not include GIS. The application of SVM to nonlinearly separable problems seems to be robust to missing data instances, errors and redundancy [5]. Thus, this thesis demonstrates the effectiveness of SVM and GIS for urban land use analysis.

The classification algorithms are designed to optimize the overall accuracy performance but for the imbalanced data good accuracy does not necessarily mean that classification of the minority classes is correct. Thus, additional performance measures like AUC, g-measure, recall have been included to study imbalance problems.

To solve the imbalance problem several techniques have been proposed and analyzed in the literature [10]. In the Real-world imbalanced datasets come from diverse application areas like medical diagnostic, fraud detection, intrusion detection, gene profiling, and object detection from satellite images. In this study the effect of three sampling techniques when applied on four large GIS datasets with an imbalance ration between 5 and 12, has been investigated. The datasets contain over a million instances each and the sampling methods considered are random sampling, under-sampling and Wilson's editing in combination with SVM.

SVM and NN were used before in various studies to predict urbanization and land cover with almost similar results, but different prediction patterns [15, 25]. Weka and Rapid Miner have been used to analyze the datasets and then the results are plotted and analyzed for accuracies and performance measures. It also seems better to break the datasets in small sizes to save time and memory so the results for files of size 5000 have been shown.

2 Literature Review

2.1 Support Vector Machines (SVMs)

SVMs developed by Vapnik [28], have gained wide acceptance because of the high generalization ability for a wide range of classification applications. Recently SVMs have shown promising performance in many applications and classifications, but they have been limited by speed particularly when the training data set is large. The hyper plane (decision boundary) constructed by SVM is dependent on only a portion of the training samples called support vectors that lie close to the decision boundary (hyper plane). They require the use of an iterative process such as quadratic programming to identify the support vectors from the labeled training set of examples. When the number of samples in the training set is huge, sometimes it is impossible to use all of them for training. Any training samples that are of no relevance to support vectors can be removed without degrading the classification results [12].

The success of SVM is very limited when it is applied to the problem of learning from imbalanced datasets in which negative instances heavily outnumber the positive instances (e.g. in gene profiling and detecting credit card fraud). Akbani, Kwek, and Japkowicz [1] studied the factors behind this failure and explained why the common strategy of under-sampling the training data may not be the best choice for SVM. They specifically chose SVM to attack the problem of imbalance data because SVM is based on strong theoretical foundations [28] and their empirical results showed that it performs well with moderately imbalanced data even without any modifications.

Since SVM only takes into account those instances that are close to the boundary, i.e. the support vectors, for building its model it is an interesting candidate for dealing with imbalanced datasets. This means that SVM is unaffected by non-noisy negative instances far away from the boundary even if they are huge in number.

SVM is considered as a useful technique for data classification. Although people consider that it is easier to use than NN, however, users who are not familiar with SVM often get unsatisfactory results at first.
A classification task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one "target value" (class labels) and several "attributes" (features). The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes.

For Data Preprocessing SVM requires that each data instance is represented as a vector of real numbers. Thus, if there are categorical attributes, we have to convert them into numeric data. Also scaling them before applying SVM is very important. The main advantage is to avoid attributes in greater numeric ranges dominate those in smaller numeric ranges. Moreover, numerical difficulties can be avoided during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems.

For the Model Selection the RBF kernel has been selected [9] out of linear, polynomial, radial base function (RBF) and the sigmoid kernel. The RBF kernel non-linearly maps samples into a higher dimensional space, so unlike the linear kernel, it can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF as [11] shows that the linear kernel with a penalty parameter $\tilde{C}$ has the same performance as the RBF kernel with some parameters (C, γ). In addition, the sigmoid kernel behaves like RBF for certain parameters [16]. The second reason is the number of hyperparameters which influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel. Moreover, it must be noted that the sigmoid kernel is not valid (i.e. not the inner product of two vectors) under some parameters [28].

However, there are some situations where the RBF kernel is not suitable. In particular, when the number of features is very large, one may just use the linear kernel.

There are two parameters while using the RBF kernels: C and γ. It is not known beforehand which C and γ are the best for one problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C,γ) so that the classifier can accurately predict unknown data (i.e., testing data). An improved version of this procedure is cross-validation.

In v-fold cross-validation, the training set is divided into v subsets of equal size. Then one subset is sequentially tested using the classifier trained on the remaining v-1 subsets. Thus, each instance of the whole training set is predicted once so the cross-validation accuracy is the percentage of data which are correctly classified. The cross-validation procedure can prevent the overfitting problem.

SVM and NN were used before in various studies to predict urbanization and land cover with almost similar results, but different prediction patterns [15, 25]. Although SVM itself does not provide a mechanism to deal with imbalanced data, it can be easily modified. SVM builds the decision boundary on a limited number of instances that are close to the boundary, being unaffected by instances far away from the boundary. This observation can be used as an active learning selection strategy that provides a balanced training set for the early training stages of the SVM algorithm.

2.2 Liblinear

If the number of features is large, one may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance. Using the linear kernel is good enough, and one only searches for the parameter C.

When Number of instances is much less than the number of features the cross-validation accuracy of using the linear kernel is comparable to that of using the RBF kernel. Thus, when the number of features is very large, one may not need to map the data. In addition to Libsvm, the Liblinear software is also effective for data in this case.

When both number of instances and features are large (as in case of document classification), Libsvm is not particularly good for this type of problems. Another software called Liblinear [17], which is very suitable for such data.

Liblinear is an open source library for large-scale linear classification [8]. It supposts logistic regression and linear support vector machines. Liblinear is very efficient for training large-scale problems. Furthermore, Liblinear is competitive with or even faster than state of the art linear classifiers. Liblinear supports two popular binary linear classifiers: LR and linear SVM.

The Liblinear package includes a library and command-line tools for the learning task. Liblinear and Libsvm share similar usage as well as application program interfaces (APIs), so users/developers can easily use both packages. However, their models after training are quite different.

The Liblinear package has a lot of documentation. The installation process, command-line usage, and the library calls are contained in the README file. Moreover, programs train.c and predict.c are good examples of using Liblinear APIs. The main design principle is to keep the whole package as simple as possible while making the source codes easy to read and maintain.

The files in Liblinear can be separated into source files, pre-built binaries, documentation, and language bindings. Liblinear can run on almost every platform as there is no dependency on external libraries and the source codes are written in C/C++. Moreover, the library calls are implemented in the file linear.cpp. The train() function trains a classifier on the given data and the predict() function predicts a given instance. Furthermore, to handle the multi-class problems via the one-vs-the-rest strategy, train()

conducts several binary classifications, each of which is by calling the train_one() function. train_one() then invokes the solver of the users' choice.

Liblinear takes much less time as compared to Libsvm and moreover, Libsvm consumes more memory. Clearly Liblinear is much faster than Libsvm to obtain a model with comparable accuracy. Liblinear is also efficient for large-scale document classification.

When Number of instances is much greater than the number of features then as the number of features is small, one often maps data to higher dimensional spaces (i.e., using nonlinear kernels). However, the linear kernel may be used with the option –s 2 in Liblinear. When the number of features is small, it is often faster than the default –s 1. Using –s 2 leads to shorter training time.

Liblinear supports L2-regularized logistic regression (LR), L2-loss and L1-loss linear SVMs [3]. Given a set of instance-label pairs (x_i, y_i), i=1,…,l, $x_i \in R^n$, $y_i \in \{-1,+1\}$, both methods (LR and linear SVM) solve the following unconstrained optimization problem with different loss functions $\xi(\omega; x_i, y_i)$:

$$\min_{\omega} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi(\omega; x_i, y_i), \tag{1}$$

where C > 0 is a penalty parameter. For SVM, the two common loss functions are

$$\max\left(1 - y_i\omega^T x_i, 0\right) \text{ and } \max\left(1 - y_i\omega^T x_i, 0\right)^2. \tag{2}$$

The former is referred to as L1-SVM, while the latter is L2-SVM.

In some applications, an SVM problem appears with a bias term *b*. This can often be dealt with by appending each instance with an additional dimension:

$$x_i^T \leftarrow \left[x_i^T, 1\right] \quad \omega^T \leftarrow \left[\omega_i^T, b\right]. \tag{3}$$

Problem (1) is often referred to as the primal form of SVM. One may instead solve its dual problem:

$$\min_{\alpha} f(\alpha) = \frac{1}{2}\alpha^T \overline{Q} \alpha - e^T \alpha \text{ subject to } 0 \le \alpha_i \le U, \forall i, \tag{4}$$

Where $\overline{Q} = Q + D$, D is a diagonal matrix, and $Q_{ij} = y_i y_j x_i^T x_j$.

For L1-SVM, U=C and $D_{ii} = 0, \forall i.$ For L2-SVM, $U = \infty$ and $D_{ii} = 1/(2C), \forall i.$

The primal L1-SVM is not differentiable whereas the primal L2-SVM is differentiable but not twice differentiable. While the dual form (4) involves bound constraints, its objective function is twice differentiable for both L1- and L2-SVM.

2.3 Neural Networks

Previous research [15, 25] has shown that classification methods such as SVM and NN can be successfully used to predict patterns of urbanization in large datasets. SVM and NN can then be used as predictive tools to determine if grid cells can be accurately predicted as urban or non-urban cells. The effectiveness of the predictive capability of the SVM and NN can be measured through standard accuracy.

Artificial NN are powerful tools that use machine learning approach to numerically solve relationships between inputs and outputs. They are designed to emulate the functionality of neurons and typically consist of many simple processing units, which are wired together in a complex communication network. Each unit or node is a simplified model of a real neuron which sends off a new signal, if it receives a sufficiently strong input signal from the other nodes to which it is connected. Depending on the activity, the strength of these connections may be varied.

One of the first neural nets was the perceptron, [23] which consists of a single node. It receives weighted inputs and thresholds the results according to a defined rule. Thus it is capable of classifying linearly separable data and performing linear functions. The Multi-layer perceptron (MLP) [24] is used widely and consists of three layers: input, hidden, and output and therefore it can be used to identify relationships that are non-linear in nature.

In the case of classification problem, neural net algorithms calculate weights for input values, input layer nodes, hidden layer nodes and output layer nodes by introducing the input in a feed forward manner. This propagates through the hidden layer and to the output layer. Then the signals propagate from node to node and are modified by weights associated with each connection. The receiving node sums the weighted inputs from all of the nodes connected to it from the previous layer. Furthermore, the output of this node is then computed as the function of its input called the "activation function." The data moves forward from node to node with multiple weighted summations occurring before reaching the output layer.

Weights in a neural network are determined by using a training algorithm, the most popular of which is the back propagation (BP) algorithm. This algorithm randomly selects the initial weights, and then compares the calculated output for a given observation with the expected output for that observation. Then the difference between the expected and calculated output values across all observations is summarized using the mean squared error. After all observations are presented to the network, the weights are modified according to a generalized delta rule [24], so that the total error is distributed among the various nodes in the network. This process of feeding forward signals and back-propagating the errors is repeated iteratively, with each iteration being called a cycle.

Presenting data to the neural net with input and output data over many cycles is called training. Users can then stop the training session and have the neural net software save all of the weights and biases to a network file. A network file is then applied to another dataset containing only input data and no output data. This process, called testing, allows the neural net to estimate output values.

NN differ from algorithm or statistical based models as they do not require formal mathematical specification. Moreover, NN are not highly sensitive to noise in data whereas statistical or mathematical algorithms treat noise in data similar to data of high quality. Lastly, NN generate information that can be applied to data that it "has not seen before." Thus, there is the potential to develop models that can be "generalized" or transferable.

Neural nets have been used with GIS and remote sensing to model urban changes [20, 22].

2.4 Imbalanced Datasets

Imbalanced training sample means that one class is represented by a large number of examples while the other is represented by only a few. Methods for reducing class imbalance in the training sample include over-sampling (replicates examples in) the minority class, and under-sampling (eliminates examples in) the majority class [2].

When classes are imbalanced, many learning algorithms can suffer from the perspective of reduced performance [10]. In binary classification, it is typically the minority (positive) class that the practitioner is interested in. Imbalance in the class distribution often causes machine learning algorithms to perform poorly on the minority class. Thus sampling techniques can be used to improve the performance of classifiers when one class is relatively rare. An efficient learning method has been proposed by Ertekin, Huang, Bottou and Giles [7] which selects informative instances from a randomly picked small pool of examples rather than making a full search in the entire training set. This strategy renders active learning to be applicable to very large datasets without sacrificing prediction performance.

Land use information provides valuable input to local, state and regional land use planning. Land transformation models can be developed to help understand what factors are important to land use change [22].

2.5 Softwares (Weka, Rapidminer and Liblinear)

2.5.1 Weka

Weka (Waikato Environment for Knowledge Analysis), is a software which was developed at the University of Waikato. It is a machine learning software written in Java that is intended to aid in the application of machine learning techniques to a variety of real-world problems.

The Weka workbench consists of a collection of visualization tools and algorithms for data analysis and predictive modelling, together with graphical user interfaces for easy access to this functionality. Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection.

Moreover, Weka provides access to SQL databases using Java Database Connectivity and it can process the result returned by a database query. Although it's main user interface is the Explorer, but it can also be accessed from the command line. It also has the Experimenter, which allows the systematic comparison of the predictive performance of Weka's machine learning algorithms on a collection of datasets.

Weka's Explorer interface is composed of several panels that give access to the main components of the weak workbench. The Preprocess panel has facilities for importing data from a database, and for preprocessing this data using a filtering algorithm. These filters can be used to transform the data (e.g., turning numeric attributes into discrete ones) and make it possible to delete instances and attributes according to specific criteria.

The Classify panel enables the user to apply classification and regression algorithms (classifiers in Weka) to the resulting dataset, to estimate the accuracy of the resulting predictive model, and to visualize erroneous predictions, ROC curves, etc. The Cluster panel gives access to the clustering techniques in Weka, e.g., the simple k-means algorithm. The Associate panel provides access to association rule learners that identify all important interrelationships between attributes in the data. The Select attributes panel provides algorithms for identifying the most predictive attributes in a dataset. The Visualize panel shows a scatter plot matrix, where individual scatter plots can be selected and enlarged, and analyzed further using various selection operators.

In some cases considerable manipulation of a database is necessary before any information can be processed by Weka. Thus, in order to maintain format independence, data is converted to an intermediate representation called ARFF (Attribute Relation File Format). ARFF files contain blocks describing relations and their attributes, together with all the instances of the relation. They are stored as plain text for ease of manipulation. Relations are simply a single word or string naming the concept to be learned. Each attribute has a name, a data type (which must be one of enumerated, real or integer) and a value range (enumerations for nominal data, intervals for numeric data).

2.5.2 RapidMiner

RapidMiner is a software used for machine learning and data mining experiments. It was formerly known as YALE (Yet Another Learning Environment). The experiments in RapidMiner can be made up of a large number of arbitrarily nestable operators which can be described in XML files created with RapidMiner's graphical user interface. RapidMiner can be accessed both from the command line and its graphical user interface.

It is written in the Java programming language and therefore can work on all popular operating systems. The buildup contains more than 400 operators for all main machine learning procedures, data preprocessing and visualization. Moreover, it also integrates all learning schemes and attributes evaluators of the Weka learning environment. Lastly, the applications of RapidMiner range from simple data mining tasks to hardcore research.

2.5.3 Liblinear

The train option in Liblinear can be used as follows

train [options] training_set_file [model_file]

The options –s1 and –s2 can be used to train the data, which are defined as

-s1= L2-loss support vector machines (dual) and -s2 = L2-loss support vector machines (primal). –s2 takes much lesser time to train the data as compared to –s1. In addition to this the option –c defines the cost parameter and -v n defines the n-fold cross validation mode. Whereas the predict option can be used in the following way

predict [options] test_file model_file output_file

Thus output files can be produced which can be compared to original data files so classify the predicted values as desired.

3 Methodology

The data for Delaware, Mahoning and Medina counties was taken in .txt format and then the train option was ran on it using Liblinear with options –s1 and –s2 in addition to –c4 and –v5 options. Two different datasets of Mahoning counties were used for this. The initial text files were broken into sizes of 25000 and 50000 and separate experiments were carried out on each of them to get the cross validation accuracies.

After this the train option was ran again to get .model files. The predict option was then used to predict the classification accuracies and the .output files were produced. These output files were read using SPSS and then merged with the original test files, after which the frequencies were found using the following SPSS script

```
IF (v1=-1 & var00001=-1) final=0.

IF (v1=-1 & var00001=1) final=1.

IF (v1=1 & var00001=-1) final=2.

IF (v1=1 & var00001=1) final=3.
```

Here 0 represents the True Negative, 1 represents the False Positive, 2 represents the False Negative and 3 represents the True Positive.

Here final was the column in which results were obtained using the first columns of the output and original files (namely v1 and var00001). The frequencies were thus obtained from the final columns which have been shown in the results.

The same method was employed for running Libsvm on the files wherein only –v5 option was used to train. The results for cross-validation accuracies, classification and frequencies were obtained.

The data for the four counties from the state of Ohio: Delaware, Holmes, Mahoning and Medina contained more than a million instances. Table 1 shows for each county dataset how many instances belong to the positive class, how many instances belong to the negative class and the ratio.

Table 1. Number of Training Instances and Ratios

Counties	*# Positive*	*# Negative*	*Ratio*
Delaware	209,765	1,106,749	5.27
Holmes	90,164	1,129,403	12.52
Mahoning	353,411	868,423	2.45
Medina	228,819	987,405	4.31

All the datasets were imbalanced from a 2.4 ratio for Mahoning county to a 12.5 ratio for Holmes county.

Three methods (LIBSVM, MYJSVM and Decision Trees) were chosen to analyze the data for Delaware, Holmes, Mahoning and Medina counties. Initially the data for these counties was available in Weka file format which was then changed to SPSS file format.

MLP was run on the weka files and accuracies, kappa statistics and confusion matrices were obtained. MLP has been a popular technique due to its capabilities to perform arbitrary mappings and not just classifications. The data was then read in Rapid Miner and then two files (.dat and .aml) were produced for each data set.

Thereafter Sampling was run on the .aml files for each county and file sizes of 5000, 10000, 20000, 50000 and 10000 were produced. As it was later realized that file of bigger size took too much time and memory, so files of size 5100-5500 were produced.

But after using this data it was realized that it was not normalized so normalization was done. A grid parameter search was performed for the SVM classifier and the values for the two parameters C and gamma were obtained.

Table 2. Accuracies, parameters using .aml files for Delaware County

Data files	Accuracy	C	Gamma
Del5000.aml	90.38%	32	0.12
Del10000.aml	90.58%	8	0.5

The above table shows the values of parameters C and Gamma which were obtained from the .aml files produced using Rapid Miner. The results have been shown for only one county as the data was normalized later.

Table 3. Parameters C and gamma for the Libsvm

Counties	C	Gamma
Delaware	8192	0.12
Holmes	2048	0.5
Mahoning	2	32
Medina	128	0.12

The above table shows the parameters C a gamma obtained using grid search.

These datasets had over a million instances and the sampling technique used for investigating the data discarded random instances from the majority class until the two classes were equally represented.

As in the case of imbalanced datasets, classification accuracy is not the best metric to valuate a classifier so performance metrics like sensitivity and specificity were used. Recall or sensitivity gives the accuracy on the positive instances.

Sensitivity= True Positive / (True Positive + False Negative)

Specificity is the accuracy on the negative instances.

Specificity= True Negative / (True Negative + False Positive)

Additionally Accuracy can be defined as

Accuracy= (True Positive + True Negative) / (True Positive + True Negative + False Positive + False Negative)

Both sensitivity and specificity are used to calculate g-means as

g-means= $\sqrt{(\text{sensitivity}*\text{specificity})}$

The classifiers SVM and MLP were trained on the 50,000 instances datasets and the models obtained were tested using the entire datasets. The results have been shown in table 14.

4 Results

An experimental analysis performed on the large imbalanced GIS extracted datasets has been presented in this section. The goal was to find out whether Liblinear works efficiently for these datasets and also to compare the performance of Liblinear with that of Libsvm. Moreover the experiments were carried with both options –s1 and –s2 in case of Liblinear and the time was noted to see the different outputs.

Additional interest was to see what outputs patterns can be produced for the land cover classification. For this values of 0 to 3 were assigned based on correct/ incorrect predicted values for corresponding initial values of urban land use.

Furthermore it was also tested if Liblinear produces the results in much lesser time than Libsvm without compromising the accuracy. Using the train and predict options both in

Liblinear and Libsvm the comparison was made on cross-validation accuracies and the percentage of correctly classified instances.

Some experiments were also carried out using Weka and Rapid miner, the results of which have also been presented. Things like accuracy, kappa statistics, values of parameters C and gamma have been calculated using these software. Finally a few results for NN have also been presented and compared with the results for SVM. Performance measures like accuracy, recall and g-means have also been calculated.

Table 4. Cross Validation Accuracies for Liblinear and Libsvm

Datasets	Libsvm Accuracy	Time (sec)	S2 Accuracy	Time (sec)
Del_25	88.96%	356.21	88.14%	2.62
Del_50	88.92%	1574.94	87.95%	4.47
DelTest			85.74%	116.59
Mah5_25	89.74%	429.24	88.54%	2.27
Mah5_50	89.71%	1195.42	88.97%	4.46
Mah5Test			86.75%	117.28
Mah8_25	85.15%	438.92	83.45%	2.26
Mah8_50	85.9%	1753.18	83.45%	4.25
Mah8Test			82.19%	101.80
Med_25	86.38%	343.44	84.94%	2.70
Med_50	86.29%	1656.61	84.62%	5.12
MedTest			83.11%	124.45

The above table shows the results for cross validation accuracies when the train option was run on the datasets using Libsvm and Liblinear respectively. Although Libsvm has slightly better cross validation accuracies than Liblinear but the time utilized by Liblinear is much less as compared to Libsvm.

Table 5. Accuracies using predict for output files for Liblinear

Datasets	S1	S2
Del25	88.18% (22047/25000)	88.19% (22049/25000)
Mah525	88.52% (22131/25000)	88.62% (22155/25000)
Mah825	83.53% (20883/25000)	83.52% (20881/25000)
Med25	84.81% (21204/25000)	84.98% (21247/25000)
Del50	87.97% (43989/50000)	87.99% (43997/50000)
Mah550	88.99% (44496/50000)	88.97% (44488/50000)
Mah850	83.47% (41735/50000)	83.47% (41738/50000)
Med50	84.46% (42234/50000)	84.61% (42307/50000)
DelTest	88.18% (1161031/1316514)	88.14% (1160470/1316514)
Mah5Test	88.55% (1081997/1221839)	88.94% (1086822/1221839)
Mah8Test	83.30% (1017819/1221839)	83.48% (1020108/1221839)
MedTest	84.63% (1029322/1216224)	84.55% (1028353/1216224)

Table 6. Accuracies using predict for output files for LIBSVM

	Accuracy (Classification)
Del25	89.04% (22262/25000)
Mah525	89.86% (22467/25000)
Mah825	85.32% (21330/25000)
Med25	86.50% (21627/25000)
Del50	88.98% (44490/50000)
Mah550	89.75% (44879/50000)
Mah850	86.07% (43038/50000)
Med50	86.36% (43183/50000)

The above two tables show the number of correctly classified instances predicted out of the total instances available during each experiment of Liblinear and Libsvm. It can be seen that Libsvm had slightly better predicted accuracies than Liblinear for the available datasets.

RESULTS FOR LIBLINEAR FILES

Table 7. Percentages for files of size 25000

Values	Del		Mah5		Mah8		Med	
	S1	S2	S1	S2	S1	S2	S1	S2
TN (0)	81.7	81.7	81.4	81.3	63.3	63.3	79.9	79.8
FP (1)	2.3	2.3	2.7	2.9	7.7	7.7	1.3	1.4
FN (2)	9.5	9.5	8.7	8.5	8.7	8.7	13.9	13.6
TP (3)	6.4	6.4	7.1	7.3	20.2	20.2	5.0	5.2

Table 8. Percentages for files of size 50000

Values	Del		Mah5		Mah8		Med	
	S1	S2	S1	S2	S1	S2	S1	S2
TN (0)	81.7	81.8	81.0	81.1	63.3	63.3	80.0	79.9
FP (1)	2.3	2.3	3.2	3.0	7.8	7.8	1.2	1.3
FN (2)	9.7	9.7	7.8	8.0	8.7	8.7	14.3	14.1
TP (3)	6.3	6.2	8.0	7.8	20.2	20.2	4.5	4.7

Table 9. Percentages for files of original size

Values	Del		Mah5		Mah8		Med	
	S1	S2	S1	S2	S1	S2	S1	S2
TN (0)	82.1	81.8	81.7	81.2	63.0	63.3	79.9	79.9
FP (1)	2.0	2.3	2.4	3.0	8.1	7.8	1.3	1.3
FN (2)	9.8	9.6	9.0	8.1	8.6	8.7	14.1	14.2
TP (3)	6.1	6.4	6.8	7.7	20.3	20.2	4.7	4.6

RESULTS FOR LIBSVM FILES

Table 10. Percentages for files of size 25000

Values	Del	Mah5	Mah8	Med
TN (0)	81.8	80.1	66.0	78.9
FP (1)	2.3	4.1	5.0	2.3
FN (2)	8.7	6.0	9.7	11.2
TP (3)	7.3	9.8	19.3	7.7

Table 11. Percentages for files of size 50000

Values	Del	Mah5	Mah8	Med
TN (0)	81.7	80.0	66.3	78.7
FP (1)	2.3	4.2	4.8	2.5
FN (2)	8.7	6.1	9.2	11.1
TP (3)	7.3	9.7	19.8	7.7

The above sets of tables show the results when the predicted values from Liblinear and Libsvm output were compared to the actual values supplied by the land use. A value of 0 was assigned to those areas where there was neither a predicted nor actual urban land use; a value of 1 was assigned to those areas where the model predicted an urban land use but there was actually a non-urban land use; a value of 2 was assigned to those areas where a non-urban land use was predicted, but the land was actually urban; and a value of 3 was assigned to those places where it was both predicted and actual urban land use.

Table 12. Accuracy and Kappa Statistic using Weka

Counties	Accuracy	Kappa Statistic	Number of Instances
Delaware	90.02	0.57	50636
Holmes	93.82	0.32	50816
Mahoning	85.80	0.64	50910
Medina	87.76	0.53	50676

For the above table the training and testing was done on the same files for each county using Weka. The accuracies are best for Delaware County while they are no so good for Mahoning County.

Table 13. Classification Performances for NN and SVM

	Mahoning		Delaware		Medina		Holmes	
	NN	SVM	NN	SVM	NN	SVM	NN	SVM
TN	65.2	67.5		81.99	78.3	78.81	92.2	92.5
FP	5.85	3.58		2.08	2.93	2.37	0.40	0.06
FN	8.43	8.55		6.80	9.53	9.95	5.78	7.08
TP	20.5	20.38		9.14	9.29	8.87	1.60	0.31

From the above table the accuracies and g-means can be calculated.

Table 14. Classification Performances for NN and SVM

		Delaware	Holmes	Mahoning	Medina
Accuracy	SVM	91.11	92.86	87.87	87.67
	MLP	80.36	93.8	85.72	87.53
Recall	SVM	57.35	4.19	70.44	47.13
	MLP	18.86	21.68	70.85	49.36
G-means	SVM	74.78	20.47	81.79	67.64
	MLP	40.02	46.46	80.63	68.97

The above table shows the results for each performance metric. It can be seen from the results obtained above that even if the SVM has higher accuracy MLP has a higher recall. Thus it's a better classification of the positive instances for three of the datasets.

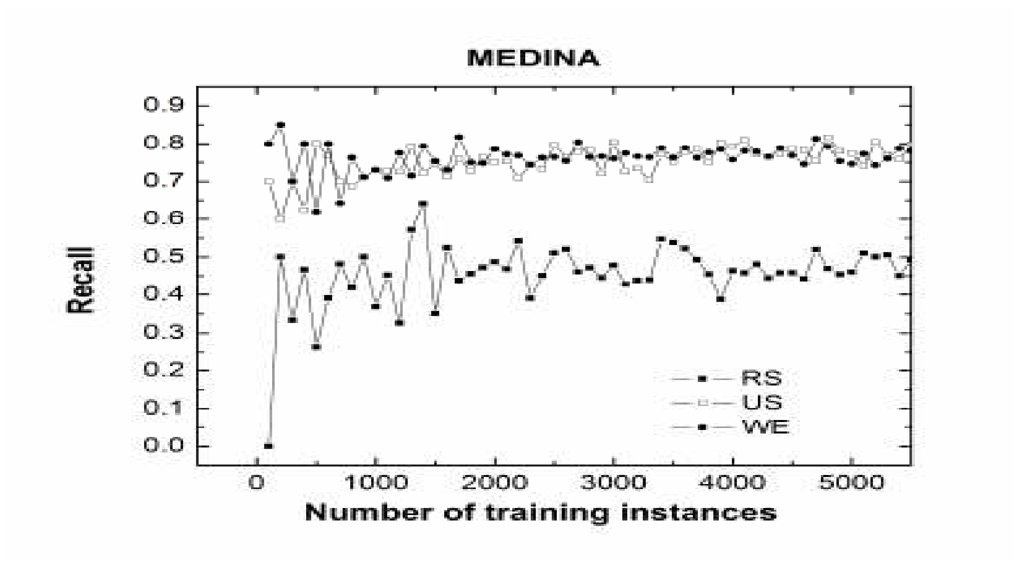

Figure 1. Recall for Medina County Dataset

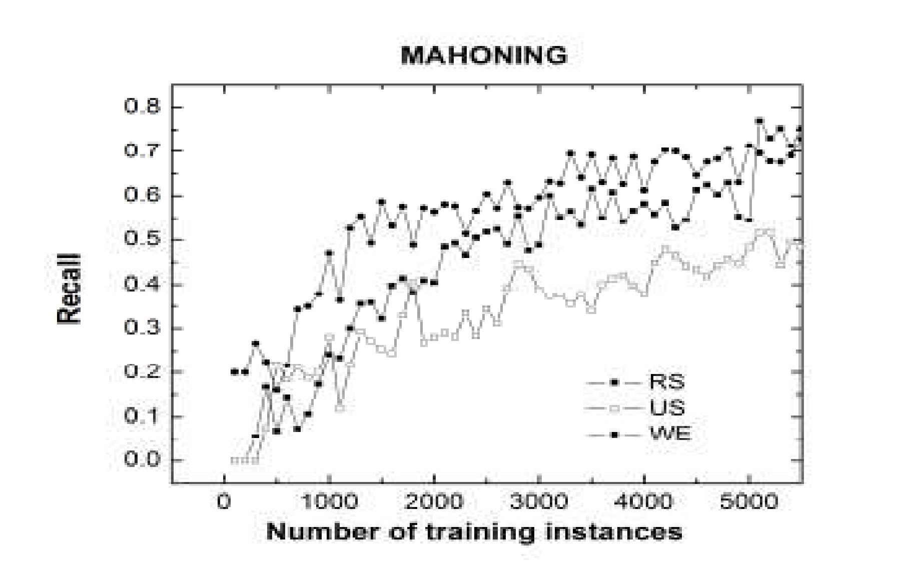

Figure 2. Recall for Mahoning County Dataset

The above figures show that both under-sampling and Wilson's editing sampling have a great influence on the classification performance of the SVM learner. As accuracy is not relevant in the case of imbalanced datasets we looked at recall and g-means. The Wilson's editing worked only slightly better than the equal under-sampling, but required extensive preprocessing.

5 Conclusion

This study has shown the effectiveness of using SVMs to predict land use which can be applied in geographic contexts with GIS. An experimental analysis on large imbalanced GIS extracted datasets has been presented here. SVMs have proved to be an effective tool for the prediction of urban areas as they deal well with large, highly non-linear datasets. Coupling, SVM prediction and classification capabilities with GIS provides a resource for determining where, spatially, over and under predictions of urbanization are occurring and allows for new modifications to be made to the model.

It can bee seen from the results that although Libsvm gives slightly better results than Liblinear but Liblinear takes much less time than Libsvm without compromising the performance. The accuracy of Liblinear is comparable to that of Libsvm and the difference is not too much in terms of accuracy. Thus, one may like to use Liblinear to save time without compromising the performance.

6 References

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. *Proc. Of European Conference on Machine Learning,* pages 39-50, 2004.

[2] R. Barandela, R. M. Valdovinos, J. S. Sanchez and F. J. Ferri, The Imbalanced Training Sample Problem: Under or Over Sampling? *In Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition(SSPR/SPR'04), Lecture Notes in Computer Science 3138,* 806-814, 2004

[3] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning*. 1992.

[4] C. Chang, and C-J Lin.

LIBSVM : a library for support vector machines, 2001. Software at http://www.csie.ntu.edu.tw/~cjlin/libsvm. Last accessed 06/15/2007.

[5] M.-W. Chang and C.-J. Lin. Leave-one-out bounds for support vector regression model selection. *Neural Computation*, 17:1188-1222, 2005.

[6] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, Cambridge, England, 2000.

[7] S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on*

Conference on information and Knowledge Management (Lisbon, Portugal, November 06 - 10, 2007). CIKM '07. ACM, New York, NY, 127-136, 2007.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification, *Journal of Machine Learning Research*, 9(2008), 1871-1874.

[9] C.-W. Hsu, C.-C. Chang, C.-J. Lin. A practical guide to support vector classification. *Technical report, Department of Computer Science, National Taiwan University*. July, 2003.

[10] J. V. Hulse, T. M. Khoshgoftaar, and A. Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international Conference on Machine Learning* (Corvalis, Oregon, June 20 - 24, 2007). Z. Ghahramani, Ed. ICML '07, vol. 227. ACM, New York, NY, 935-942, 2007.

[11] S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667-1689, 2003.

[12] R. Koggalage, and S. Halgamuge. "Reducing the Number of Training Samples for Fast Support Vector Machine Classification." *Neural Information Processing – Letters and Reviews* 2 (3), 2004, pp. 57-65.

[13] M. Kubat, R. C. Holte, and S. Matwin. Machine Learning for the detection of oil spills in satellite radar images. *Machine Learning,* 30(2-3):195-215, 1998.

[14] A. Lazar, "Income Prediction via Support Vector Machines" in Proceedings of ICMLA'2004: *The 2004 International Conference on Machine Learning and Applications, IEEE*, 2004.

[15] A. Lazar and B. Shellito. Comparing Machine Learning Classification Schemes – a GIS Approach. In *Proceedings of ICMLA'2005: The 2005 International Conference on Machine Learning and Applications,* IEEE, 2005.

[16] K.-M. Lin and C.-J. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449-1559, 2003. URL http://www.csie.ntu.edu.tw/~cjlin/papers/rsvmTEX.pdf

[17] C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research*, 9:627-650, 2008. URL http://www.csie.ntu.edu.tw/~cjlin/papers/logistic.pdf . Software available

at http://www.csie.ntu.edu.tw/~cjlin/liblinear .

[18] Y. Liu, A. An, and X. Huang. Boosting Prediction Accuracy on Imbalanced Datasets with SVM Ensembles.

[19] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler**,** YALE: Rapid Prototyping for Complex Data Mining Task. *In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06),* 2006.

[20] B. Pijanowski, S. Pithadia, B. Shellito, and K. Alexandridis. "Calibrating a neural network based urban change model for two metropolitan areas of the upper Midwest of

the United States. "*International Journal of Geographical Information Science*. 19 (2), 2005, pp. 197-216.

[21] B. Pijanowski, D. Brown, B. Shellito, and G. Manik. "Use of Neural Networks and GIS to Predict Land Use Change." *Computers, Environment, and Urban Systems*. 26(6), 2002, pp. 553-575.

[22] B. Pijanowski, B. Shellito, M. Bauer, and K. Sawaya. "Using GIS, Artificial Neural Networks and Remote Sensing to Model Urban Change in the Minneapolis-St. Paul and Detroit Metropolitan Areas." In: *Proceedings of the ASPRS Annual Conference*, St. Louis, MO, 2001.

[23] F. Rosenblatt 1958 The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **65**: 386±408

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams (1986). *Parallel Distributed Processing*. Cambridge, MA. MIT Press.

[25] B. Shellito and A. Lazar, "Applying Support Vector Machines and GIS to Urban Pattern Recognition," in *Papers of the Applied Geography Conferences*, volume 28, 2005.

[26] B. Shellito, and B. Pijanowski. "Using Neural Nets to Model the Spatial Distribution of Seasonal Homes." *Cartography and Geographic Information Science* 30 (3), 2003, pp. 281-290.

[27] B. Schölkopf and A. Smola, *Learning with Kernels*. MIT Press, Cambridge Massachusetts, 2002.

[28] V. N. Vapnik, *The Nature of Statistical Learning Theory,* 2nd edition, Springer-Verlag, New York, NY, 1999.

[29] I.H. Witten, and E. Frank, *Data Mining -Practical Machine Learning Tools and Techniques with Java Immplementation*, Morgan Kaufmann Publishers, 2000.

[30] X. Zhu. Lazy Bagging for Classifying Imbalanced Data. In *Seventh IEEE International Conference on Data Mining* (Omaha, NE, 28-31 October, 2007), 763-768, 2007

[31] Census 2000. 2000. *Census Factfinder*. US Census Bureau. http://factfinder.census.gov. Last accessed June 13, 2005.