Aerial LiDAR Data Classification using Support Vector Machines (SVM)

Suresh K. Lodha Edward J. Kreps David P. Helmbold Darren Fitzpatrick
Department of Computer Science
University of California, Santa Cruz, CA 95064
{lodha,jay,dph,darrenf}@soe.ucsc.edu

Abstract

We classify 3D aerial LiDAR scattered height data into buildings, trees, roads, and grass using the Support Vector Machine (SVM) algorithm. To do so we use five features: height, height variation, normal variation, LiDAR return intensity, and image intensity. We also use only LiDAR-derived features to organize the data into three classes (the road and grass classes are merged). We have implemented and experimented with several variations of the SVM algorithm with soft-margin classification to allow for the noise in the data. We have applied our results to classify aerial LiDAR data collected over approximately 8 square miles. We visualize the classification results along with the associated confidence using a variation of the SVM algorithm producing probabilistic classifications. We observe that the results are stable and robust. We compare the results against the ground truth and obtain higher than 90% accuracy and convincing visual results.

Keywords: LiDAR data, classification, Support Vector Machine (SVM), terrain, visualization.

1. Introduction

Aerial and ground-based LiDAR data is being used to create virtual cities [25, 10, 16], terrain models [23], and classify different vegetation types [3]. Typically, these datasets are quite large and require some sort of automatic processing. The standard technique is to first normalize the height data (subtract a ground model), then use a threshold to classify data into into low- and high-height data. In relatively flat regions which contain few trees this may yield reasonable results, e.g. the USC campus [25]; however in areas which are forested or highly-sloped, labor-intensive manual input and correction is essential with current methods in order to obtain an useful classification.

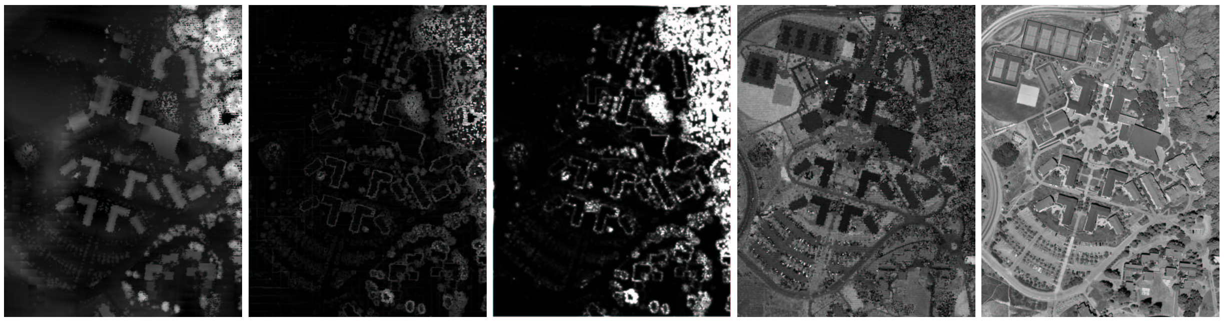
In this work, we have used Support Vector Machines (SVM) for automatic classification of aerial LiDAR data registered with aerial imagery into four classes – buildings, trees (or high vegetation), roads, and grass. In cases where aerial imagery is not available, our algorithm classifies aerial LiDAR data automatically into three classes – buildings, trees, and road-grass.

We have experimented with a number of parameters associated with the use of the SVM algorithm that can impact the results. These parameters include choice of kernel functions, the standard deviation of the Gaussian kernel, relative weights associated with slack variables to account for the non-uniform distribution of labeled data, and the number of training examples. We discuss the algorithm and these parameters in Section 4. We have also used an extension of the SVM algorithm to multiclass problems that allows probabilistic classification. This helps in assigning confidences to the predicted classes. We discuss this extension in Section 4 as well.

We have used SVM to classify aerial LiDAR data collected over an eight square mile region of the UCSC campus. We have conducted a large number of tests with various parameters and observed that the results are stable and robust. We present our results in Section 5. We have visualized the classification results for several different subregions (parts of which are manually labeled for training and accuracy assessment) as well as for the whole region. We compare the results against the ground truth and obtain higher than 90% accuracy and convincing visual results. Furthermore, visualization of the classification predictions with their associated confidences (or uncertainty) allows us to investigate and understand the weak spots of the algorithm or data. Conclusions and future directions are presented in Section 6.

2. Previous Work

Several algorithms for classifying data into terrain and non-terrain points have been presented including those by Kraus and Pfeifer [12] using iterative linear prediction scheme, by Vosselman et. al. [23] using gradient-based techniques, and by Axelsson [2] using adaptive irregular triangular networks. Sithole and Vosselman [19] present a



(a) Height (H) (b) Height Variation (HV) (c) Normal Variation (NV) (d) LiDAR Return Intensity (LRI) (e) Image Intensity (I)

Figure 1. The five features used in data classification for one of the training regions.

comparison of these algorithms. We have used a variation of these standard algorithms to create a terrain model from aerial LiDAR data in order to compute normalized height.

The objective of this work is to perform classification of aerial LiDAR data into 4 classes using all the 5 features or 3 classes using only 3 height-derived features. Previous classification efforts include research by Axelsson [2], Maas [13], Filin [9], Haala and Brenner [11], and Song et. al. [20]. Most of these approaches are ad-hoc heuristics. Also, in most cases results are presented for small regions without much discussion on the quality of results obtained. Finally, these approaches often require substantial manual input and sometimes tweaking of weight parameters.

Anguelov et al. [1] classify ground-based LiDAR scan points as building, tree, or shrub using a Markov Random Field model that jointly classifies data points while encouraging spatial contiguity. They report an accuracy rate of 93% for their associative Markov networks algorithm, and 68% to 73% accuracy for SVM methods on their data.

Previous work on the same aerial LiDAR data used the EM (Expectation-Maximization) algorithm with a mixture of Gaussian models [7] to classify LiDAR data into four categories. There are several differences between this previous work and our current work. The previous work required additional information – DEM data, co-registered with the LiDAR data. Also, here we use a different feature set: we introduce normal variation, which we find very useful in our classification. Finally, as reported in Section 5, we obtain higher accuracy (better than 93%) in comparison to the 66-84% accuracy reported in the previous work.

3. Data Processing

3.1 Data Collection and Preparation

Our LiDAR dataset covers 8 square miles of the campus at the University of California at Santa Cruz and was collected using a 1064 nm laser at a pulse rate of 25 KHz. The raw data consists of about 36 million points with an average point spacing of 0.26 meters. We resampled this irregular LiDAR point cloud onto a regular grid with a cell size of 0.5m using nearest-neighbor interpolation. In addition, we used high resolution (0.5ft/pixel) ortho-rectified grayscale aerial imagery. To match the LiDAR data these images are downsampled to 0.5m/pixel, then registered using the NAD83 State Plane Coordinate System, California Zone III.

3.2. Features

We identified five features to be used for data classification purposes: normalized height, height variation, normal variation, LiDAR return intensity, and image intensity.

- *Normalized Height (H)*: We computed the terrain elevation data automatically from the aerial LiDAR data using a variation of the standard DEM extraction algorithms [19]. The LiDAR data is normalized by subtracting terrain elevations from the LiDAR data.
- *Height Variation (HV)*: Height variation is measured within a 3×3 pixel $(2.25m^2)$ window and is calculated as the absolute difference between the min and max values of the normalized height within the window.
- *Normal Variation (NV)*: We first compute the normal at each grid point using finite differences. Normal variation is the average dot product of each normal with

other normals within a 10×10 pixel $(25m^2)$ window. This value gives a measure of planarity within the window.

- *LiDAR Return Intensity (LRI)*: Along with height values, aerial LiDAR data contains the amplitude of the response reflected back to the laser scanner. We refer to this amplitude as LRI.
- *Image Intensity (I)*: Image intensity corresponds to the response of the terrain and non-terrain surfaces to visible light. This is obtained from the gray-scale aerial image.

Figure 1 shows images of these five features computed on the College 8 Region of the UCSC Campus. The feature values were compressed to byte values (0-255) to ease storage requirements.

3.3. Classes and Training

We classified the dataset into 4 classes: roofs (or buildings), trees or high vegetation (includes coniferous and deciduous trees), grass (includes green and dry grass), and roads (asphalt roads, concrete pathways and soil). When using only three height-derived features H, HV, and NV, we classified the data into 3 classes by merging roads and grass into the same class. Datasets for ten different regions were segemented and some subregions were manually labeled for training and validation. The size of these labeled data sets vary from 100,000 points to 150,000 points. The mix of different classes – trees, grass, roads, and roofs – vary within these data sets. Roughly 25-30% of these data sets were labeled to cover these 4 classes adequately.

4. The SVM Algorithm

Support vector machines (SVMs) are a powerful learning method that is widely used in a variety of domains. Originally introduced by Boser, Guyon and Vapnik [4], SVMs and related kernel-based methods are the subject of a large body of work in classification, regression, and clustering problems. We give only a cursory summary of the algorithm and refer the reader to any of the excellent introductions to the topic for more details (for example, Schölkopf and Smola [18], Christianini and Shaw-Taylor [8], or Burges [5]). We first outline the SVM algorithm in the most basic binary classification setting and then discuss the necessary extensions for more complex prediction problems, including the LiDAR classification problem.

4.1. Maximum Margin Classification

The problem of binary classification is to use a set of training data to create a classifier which will correctly predict the label of future examples that are not present in the training set. Let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\} \subseteq \mathcal{X} \times \mathcal{Y}$ be the m-example training set where each $\mathbf{x}_i$ is an n dimensional feature vector and y_i is an integer class label. We need only consider the case where $\mathcal{X} = \Re^n$; for binary classification we use the convention that $\mathcal{Y} = \{-1, 1\}$.

A binary classifier is a function $h : \Re^n \mapsto \{-1, 1\}$. Among the simplest possible such functions are the linear classifiers $h_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$. If there exists $\mathbf{w}$ and b such that $h_{\mathbf{w},b}(\mathbf{x}_i) = y_i$ for $i = 1, \ldots, m$ then the training set is *linearly separable*.

The *geometric margin* measures the distance from examples to the separating hyperplane of a hypothesis. The geometric margin of example $(\mathbf{x}, y)$ with respect to hypothesis $h_{\mathbf{w},b}$ is $\gamma_{\mathbf{w},b}(\mathbf{x}, y) = y(\frac{\mathbf{w}\cdot\mathbf{x}+b}{\|\mathbf{w}\|})$.

If $\mathbf{x}$ is incorrectly classified, then the margin is negative, and $-\gamma_{\mathbf{w},b}(\mathbf{x}, y)$ is the distance to the hyperplane. The geometric margin of a sample $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ with respect to some $h_{\mathbf{w},b}$ is the minimum margin of any of the examples: $\gamma_{\mathbf{w},b}(S) = \min_i \gamma_{\mathbf{w},b}(\mathbf{x}_i, y_i)$.

Any $h_{\mathbf{w},b}$ achieving the maximum geometric margin on a linearly separable training set is called a *maximum margin hypothesis*. Statistical learning theory provides evidence that maximum margin hypotheses are likely to generalize well to new data.

The problem of finding a maximum margin hypothesis for a linearly separable data set is equivalent to the constrained minimization problem:

$$\underset{\mathbf{w},b}{\text{minimize}} \ \frac{\|\mathbf{w}\|^2}{2} \tag{1}$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \text{ for } i = 1, ..., m.$$

That is, we minimize the length of $\mathbf{w}$ (and thus maximize the margin) subject to the constraint that all points have distance at least 1 from the separating hyperplane.

The dual of this optimization problem is

$$\underset{\alpha\in\Re^m}{\text{maximize}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \tag{2}$$

$$\text{subject to each } \alpha_i \geq 0, \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

The dual form is a quadratic programming problem to which well known algorithms are applicable. In practice a specialized algorithm called Sequential Minimal Optimization (SMO) algorithm [15] is preferred due to its efficiency.

4.2. Extending the maximum margin

The simple maximum margin framework as outlined above has three significant drawbacks which must be overcome before it can be applied to difficult classification problems like ours. First, the decision boundary of the resulting classifier, h, must be a linear combination of the features; second, the training set must be linearly separable; and third, only binary classification is possible. The SVM literature contains numerous solutions to these problems.

High-dimensional embedding with kernels:

Even if the data is not linearly separable with the original features, it may become linearly separable if new non-linear features are constructed. Thus instead of working directly with feature vector $\mathbf{x}$ as above, we can apply a non-linear transformation Φ mapping $\mathbf{x}$ to $\Phi(\mathbf{x})$ in a (usually much) higher dimensional space, and attempt to find a maximum margin separating plane in the higher dimensional space.

Note that the dual form (2) depends on the feature vectors only through a dot product. Therefore a *kernel function* K such that $K(x_i, x_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ can be used to avoid the computational expense of working in the high dimensional space. Kernel functions operate directly on the original features and only implicitly represent the transformation to the high-dimensional space. Two popular kernel functions are:

degree-d polynomial	$K(x_i, x_j) = (x_i \cdot x_j + c)^d$
Gaussian (RBF)	$K(x_i, x_j) = \exp(-\frac{1}{2\sigma}\|\mathbf{x} - \mathbf{x}_i\|^2)$

Soft-margin classification:

Even after transformation to a higher dimensional space, outliers may prevent the data from being linearly separable. In the soft-margin technique each example $(\mathbf{x}_i, y_i)$ has a slack variable $\xi_i \geq 0$ that measures how much the margin for that example falls below the desired unit distance. The primal minimization problem (1) is modified to minimize both the length of $\mathbf{w}$ and the sum of the slack variables:

$$\underset{\mathbf{w},b,\xi}{\text{minimize}} \ \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{m}\sum_{i=1}^{m} \xi_i$$

$$\text{subject to } y_i(K(\mathbf{x}_i, \mathbf{w}) + b) \geq 1 - \xi_i, i = 1, \ldots, m,$$

where C trades off the relative importance of slack variable minimization and margin maximization.

Imbalanced data:

Many classes of geographical data may be important but rare. A classifier which misses rare classes can have high accuracy but undesirable behavior. This effect was quite pronounced in our own data where, for some regions, rare classes represent only 2% of the labeled data for the region.

We thus consider two different measures of accuracy for our classifier. The *sample-weighted accuracy* is simply the percentage of correctly classified points, this corresponds directly to the usual use of the term "accuracy". We also consider what we term the *class-weighted accuracy* which splits the points into classes, computes the percentage of each class that is correctly classified, and averages these percentages. For example, consider a region that is 90% grass and 10% road and a classifier that correctly classifies 90% of the grass points but only 30.0% of the road points. This classifier has a sample-weighted accuracy of 84% but a class-weighted accuracy of only 60% on this region.

In our early experiments, we observed that the $\frac{C}{m}\sum_{i=1}^{m} \xi_i$ term in the objective function tends to maximize the sample-weighted accuracy at the expense of class-weighted accuracy. Our solution is to modify the slack variable term so that the under-represented class is taken more seriously. We use a set of class-specific parameters $C_1, \ldots, C_k$ in the following oprimization problem:

$$\underset{\mathbf{w},b,\xi}{\text{minimize}} \ \frac{\|\mathbf{w}\|^2}{2} + \frac{1}{m}\sum_{i=1}^{m} C_{y_i}\xi_i$$

$$\text{subject to } y_i(K(\mathbf{x}_i, \mathbf{w}) + b) \geq 1 - \xi_i, i = 1, \ldots, m.$$

The special case where $C = C_1 = \ldots = C_k$ corresponds to the previous formulation.

We must now choose each C_i. Choosing a single C is difficult to do *a priori*, and Schoelkopf et al. propose a method to avoid doing so [17]. For our own problem we found that the the overall parameter C (or the sum of all C_i) made little difference. The relative weighting between classes, however, could make a large difference in class-weighted accuracy for regions in which the proportion of classes in the training set was different from the test set. We hold out some data from the training set to estimate the proportions of the various classes and then set the outlier penalty weights C_i for each class to be inversely proportional to the frequency of the class in the training set. This essentially weights the training set so that each class is equally well represented.

Multiple Classes:

The classifier described thus far distinguishes between two classes only; however, geographical regions are often most naturally described by more than two classes. Numerous methods are known for translating SVMs (or other binary classifiers) into a multiclass setting, including: (a) training a classifier to distinguish each class from all other classes, commonly called "One vs. All", (b) training a classifier to distinguish between each pair of classes, or (c) using some extension of the concept of the margin to include more than two classes, and performing optimization directly on this quantity. We use a variation on (b) proposed by Wu et al. [24] which allows us to provide probability estimates for each class rather than a single hard classification.

Our methodology is as follows: For each pair of classes we train an SVM on data from only those two classes. Each of these SVMs is modified so that it predicts a pair of probabilities for its two classes rather than just a single class label. When predicting the label of a new point, the algorithm takes the pairwise class probabilities from the trained SVMs and then merges them into a single distribution over all of the classes using the optimization procedure of Wu et al. [24]. The class with the highest probability in this merged distribution is the final prediction.

We use a modification of the technique proposed by Platt [14] to produce pairwise probability estimates for each of the trained SVMs. Given a road-tree classifier $h_{\mathbf{w},b}$ that computes $f_{\mathbf{w},b}(\mathbf{x})$ and predicts road (R) if $f_{\mathbf{w},b}(\mathbf{x}) > 0$ and tree (T) otherwise, Platt creates a function $p_{R,T}(\mathbf{x})$ estimating $P(y = R|\mathbf{x}, y = R \text{ or } y = T)$ by fitting the following logistic sigmoid to the data

$$p_{R,T}(\mathbf{x}) = \frac{1}{1 + \exp(Af_{\mathbf{w},b}(\mathbf{x}) + B)}.$$

The parameters A and B are fit by minimizing the negative log likelihood of the the road-tree training data:

$$\sum_{i=1}^{m} \rho(y_i) \log(p_{R,T}(\mathbf{x}_i)) + (1 - \rho(y_i)) \log(1 - p_{R,T}(\mathbf{x}_i)) \tag{3}$$

where $\rho(R) = 1$ and $\rho(T) = 0$ reflect the class labels.

Platt points out that this method has a bias problem as the same data is used to both train the SVM and to fit the sigmoid. We follow his suggestion to regularize, setting $\rho(R) = \frac{N_R+1}{N_R+2}$ and $\rho(T) = \frac{1}{N_T+2}$ where N_R is the number of road points in the data and N_T is the number of tree points.

We found that a further modification was needed to protect the probabilities of rare classes. Therefore we adjust the likelihood function (3) by multiplying each term in the summation by C_{y_i} (recall that C_{y_i} is inversely proportional to the frequency of class y_i in the training data). This correction essentially causes the likelihood calculation to act as if the two classes were represented equally in the training set.

5. Results and Discussion

We use a modified version of the freely available LIBSVM [6] to test the performance of the SVM algorithm and its probablistic multiclass extensions on our LiDAR data. Figure 2 presents the accuracy results for the ten training regions using the leave-one-out test (train on the labeled subset of nine regions and test on the labeled subset of tenth region) using a radial basis function (RBF) kernel. Our RBF kernels typically used $\sigma^2 = 0.01$ as the variance of the Gaussian and C_i as inversely proportional to the frequency of that class in the given training data set as described before in Section 4.2. We have used both sample-weighted and class-weighted accuracy (described under "*Imbalanced data*" in Section 4.2) to assess the results.

We obtained better than 90% sample-weighted accuracy for all regions and better than 90% class-weighted accuracy for 8 out of 10 regions for 5 feature 4-way classification. The other two regions have a severe data imbalance (very few road points) that makes good class-weighted accuracy a challenge. The results for 3 feature 3-way classification are better than 94%.

Region	5 features, 4 classes		3 feature, 3 classes	
	class-wt accuracy %	sample-wt accuracy %	class-wt accuracy %	sample-wt accuracy %
Reg. 1	74.26	93.13	86.61	95.40
Reg. 2	92.32	94.82	98.09	97.84
Reg. 3	94.90	95.50	97.14	96.05
Reg. 4	93.90	96.21	96.54	97.95
Reg. 5	94.38	95.27	96.22	96.39
Reg. 6	92.20	90.53	96.25	98.52
Reg. 7	91.68	91.19	95.66	96.92
Reg. 8	92.07	94.94	97.90	96.52
Reg. 9	73.95	90.90	86.17	94.55
Reg. 10	95.10	95.31	98.83	98.91
Average	89.48	93.78	94.94	96.91

Figure 2. Accuracy by region using the RBF (Gaussian) kernel. For each region, training was done on 150,000 points chosen at random without replacement from nine of the regions and testing was done on all labeled points in the remaining region.

Previous work on the same LiDAR data using Gaussian mixture models achieved accuracy rates of 66–84% [7]. Anguelov et al. use SVMs as well as a graph cuts based algorithm to classify ground-based LiDAR terrain data [1]. They report 68% to 73% accuracy using SVM methods and 93% accuracy for their spatially coherent associative Markov Networks algorithm. Since their LiDAR data has very different properties from ours, we are reluctant to draw conclusions from these accuracy numbers.

Figure 3 reorganizes our accuracy results to identify the two types of error – misclassified points by correct class (Error I) and misclassified points by prediction (Error II). These errors are less than approximately 10% for all classes for 3-way classification but higher than 10% for road and grass for 5-way classification. Road and Grass prove to be the most difficult classes to adequately seperate. This is understandable as their height information is relatively similar in many areas.

We have performed extensive testing in order to determine how robust the algorithm is to changes in parameters. These include experiments with various kernel functions and their parameters, as well as more general parameters such as variations in training data size and C, which trades between margin maximization and slack variables.

Figure 4 reports the results from experiments using various kernel functions. Although the Gaussian kernel was our choice and shows marginally superior performance, it is worth noting that the linear and sigmoid kernels also perform well. Indeed, linear kernels can produce far sparser kernels and thus train and classify much more quickly. This shows that it is primarily maximizing the margin rather than the high-dimensional embedding that gives the SVM high

5/4	Tree	Grass	Road	Bldg.	Error I
Tree	97.03	0.37	0.65	1.95	2.97
Grass	2.31	78.31	11.91	7.47	21.69
Road	0.87	8.94	89.94	0.25	10.06
Building	4.27	2.53	0.58	92.63	7.38
Error II	7.44	11.83	13.14	9.68	

3/3	Tree	Road-Grass	Bldg.	Error I
Tree	96.17	0.77	3.05	3.83
Road-Grass	1.18	92.45	6.37	7.56
Building	3.42	0.38	96.21	3.79
Error II	4.60	1.15	9.42	

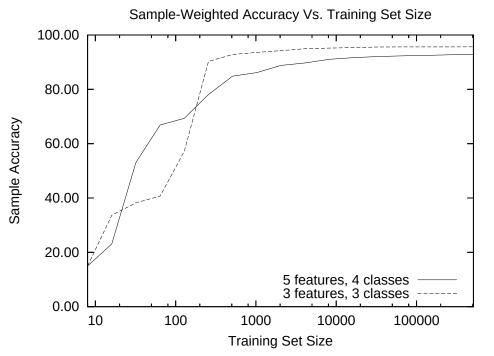
Figure 3. Confusion matrices for 4 classes/5 features and 3 classes/3 features for the results presented in Figure 2. The vertical axis is the true class, and the horizontal axis is the predicted class. Type I errors are points of a given class misclassified as another; Type II errors are points misclassified as the given class.

5 features, 4 classes		
Kernel Type	sample-weighted accuracy (%)	class-weighted accuracy (%)
RBF	93.78	89.48
Linear	90.67	91.56
Sigmoid	90.20	91.18
Polynomial deg. 2	89.03	90.01
Polynomial deg. 3	75.71	80.62
Polynomial deg. 4	49.90	63.66
Polynomial deg. 5	37.47	32.91
Polynomial deg. 6	46.63	28.92

Figure 4. Classification accuracy using different kernels: Training is done on a 100,000 point random sample from all geographical regions and testing is done on a similarly drawn disjoint sample of equal size.

accuracy on our data. For our chosen Gaussian kernel, we also experimented with various values of σ, which determines the kernel width. As shown in the lower plot of Figure 5, performance is quite stable under variations of σ.

The upper diagram of Figure 5 shows that acceptable accuracy is achieved even with relatively small training sets. Performance also remained quite stable under changes in C, the relative weighting between margin maximimization and the slack variables (the top plot in Figure 6) so long as it remained higher that 0.1. The stability under variations in the individual C_i values is slightly more complicated. In most published work only a single value $C = \sum_{i=1}^{k} C_i$ is used for all classes. Both class- and sample-weighted accuracy suffered when the relative proportions between the values C_i and the class proportions in the test set differed significantly (see the bottom plot in Figure 6).



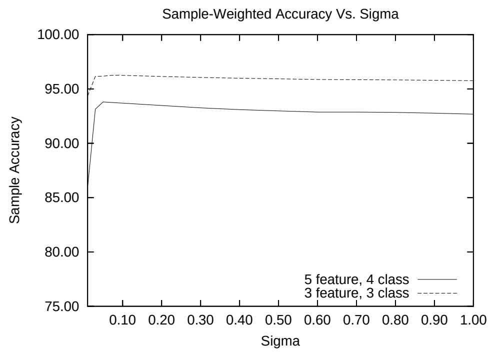


Figure 5. (Top): Sample-weighted accuracy as a function of training set size. (Bottom): Reasonable values of σ (Gaussian kernel width) all give comparable results.

We show details of the visualization of classification results for one of the regions (see Figure 7). This visualization brings out the buildings (in blue), trees (in green), roads (in brown), and grass (in yellow). For the purpose of visualizing classification confidence we use the ratio of the difference between the most and next-most probable and the most probable classification to determine the saturation of each point. Thus darker points are classified with lower confidence. Areas around the edges of buildings prove to be difficult to classify. These areas have atypical coloring, and high height and normal variance, making them easily confused with trees. Cars, shrubs, and other objects for which we have no fully adequate class also frequently occur. In general we noticed that the difficulties of the classifier tend to follow the difficulties of a human classifier. Indeed many misclassifications represent truly ambiguous regions, e.g. trees which overhang buildings, or small patches of what seems to be grass or shrubs in the middle of a forested region.

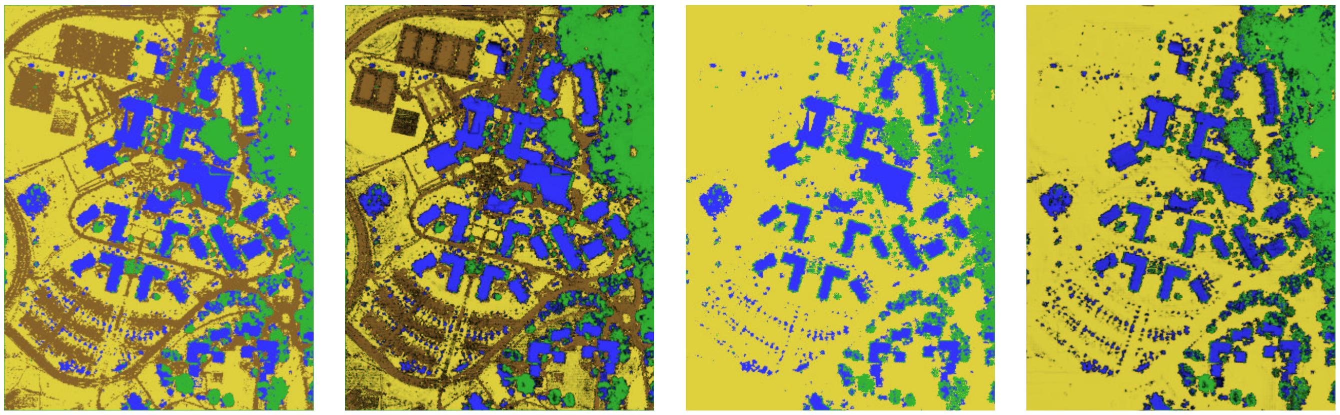


Figure 7. (leftmost) Four-way classification: buildings (blue), trees (green), roads (brown), and grass (yellow); (2nd from left) Four-way confidence rated classification (darker means less confidence); (3rd from left) Three-way classification (buildings (blue), trees (green), roads and grass (yellow); (rightmost) Three-way confidence rated classification of the same region. Training for this test was done on a training set of size 150,000 points drawn randomly from the nine other regions.

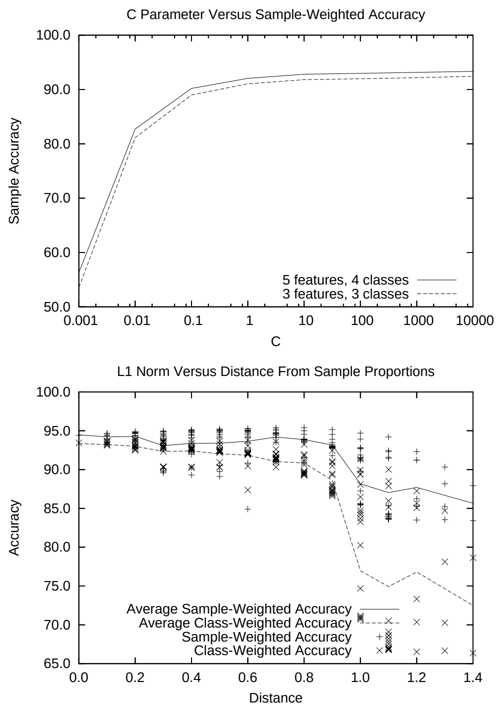


Figure 6. (Top): Sample-weighted accuracy as a function of overall $C = \sum_{i=1}^{k} C_{y_i}$. (Bottom): Accuracy as a function of relative entropy to the sample distribution. As proportions differ greatly from the training set accuracy declines.

6. Conclusions and Future Directions

We have applied the SVM algorithm to classify aerial LiDAR data and established that stable, robust, accurate classification results are achievable. Still, there are several areas of improvement worth exploring. First, spatial coherence may be used to improve the results further. One could perhaps use recently developed large-margin SVM-like algorithms which can be kernelized and work directly with a statistical model using an extended concept of the margin [21] [22]. In recent work Zabih and Kolmogorov [26] propose a segmentation algorithm that simultaneously uses both feature space and geographic space to cluster pixels and Anguelov et al. [1] learn a model that jointly classifies data points while encouraging spatial coherency. Although considering the classifications and confidences of nearby points is a powerful technique to reduce noise and outliers, this must be done very carefully in our application. Since we are interested in eventually recovering building footprints, the "rounding" of corners could create problems. Furthermore, important details like thin road segments and small isolated buildings could be missed if one aggressively enforced spatial coherency. Second, other machine learning algorithms such as AdaBoost may also be used to obtain better or perhaps simpler and more insightful classification of aerial LiDAR data. We plan to explore these approaches in our future work.

Acknowledgements

This research is partially supported by the Multi-disciplinary Research Initiative (MURI) grant by U.S. Army Research Office under Agreement Number DAAD19-00-1-0352, the NSF grant ACI-0222900, and the NSF-REU grant supplement CCF-0222900.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3D range data. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, California, June 2005.

[2] P. Axelsson. Processing of laser scanner data -algorithms and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):138–147, 1999.

[3] J. B. Blair, D. L. Rabine, and M. A. Hofton. The laser vegetation imaging sensor: a medium altitude, digitsation-only, airborne laser altimeter for mapping vegetaion and topography. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:115–122, 1999.

[4] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learing Theory*, pages 144–152, 1992.

[5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[7] A. P. Charaniya, R. Manduchi, and S. K. Lodha. Supervised parametric classification of aerial lidar data. *IEEE workshop on Real-Time 3D Sensors*, pages 25–32, June 2004.

[8] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods).* CUP, 2000.

[9] S. Filin. Surface clustering from airborne laser scanning data sagi filin. In *ISPRS Commission III, Symposium 2002 September 9 - 13, 2002, Graz, Austria*, pages A–119 ff (6 pages), 2002.

[10] C. Frueh and A. Zakhor. Constructing 3D city models by merging ground-based and airborne views. In *IEEE Conference on Computer Vision and Pattern Recognition 2003, June 2003*, 2003.

[11] N. Haala and C. Brenner. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):130–137, 1999.

[12] K. Kraus and N. Pfeifer. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53:193–203, 1998.

[13] H.-G. Maas. The potential of height texture measures for the segmentation of airborne laserscanner data. *21st Canadian Symposium on Remote Sensing, Ottawa, Ontario, Canada*, 1999.

[14] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74, 2000.

[15] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods: support vector learning*, pages 185–208, Cambridge, MA, USA, 1999. MIT Press.

[16] W. Ribarsky, T. Wasilewski, and N. Faust. From urban terrain models to visible citites. *IEEE Computer Graphics and Applications*, 22(4):10–15, July 2002.

[17] B. Schölkopf, A. Smola, R. Williamson, and P. Bartlett. New support vector algorithms, 2000.

[18] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[19] G. Sithole and G. Vosselman. Comparison of filtering algorithms. In *ISPRS Commission III, Symposium 2002 September 9 - 13, 2002, Graz, Austria*, 2002.

[20] J. H. Song, S. H. Han, K. Y. Yu, and Y. I. Kim. A study on using lidar intensity data for land cover classification. In *ISPRS Commission III, Symposium 2002 September 9 - 13, 2002, Graz, Austria*, 2002.

[21] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[22] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Twenty-first international conference on Machine learning*, New York, NY, USA, 2004. ACM Press.

[23] G. Vosselman. Slope based filtering of laser altimetry data. *International Archives of Photogrammetry and Remote Sensing*, 33, 2000.

[24] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[25] S. You, J. Hu, U. Neumann, and P. Fox. Urban site modeling from lidar. In *Second International Workshop on Computer Graphics and Geometric Modeling*, 2003.

[26] R. Zabih and V. Kolmogorov. Spatially coherent clustering using graph cuts. In *CVPR*, pages 437–444, 2004.