K-Means and PCA for Image Segmentation

Daniel Hanson, Sam Kreter, Brendan Marsh, and Christina R.S. Mosnick

***Abstract*—Handwriting recognition allows for handwritten documents to be stored in electronic form. Some benefits of having documents in electronic form include the reduction of space needed to store the documents and the ability to process the handwritten data within these documents. This paper proposes a method for processing the handwritten data using support vector machines (SVMs). The implementation uses 2-class classification SVMs with a multiclass method to provide a robust method for classifying handwritten characters.**

***Index Terms*—PCA, K-Means, Normalization, Whitening, Spectral Imaging.**

I. Introduction

UNSUPERVISED clustering methods have applications in many fields, such as marketing, city planning, and insurance[reference]. For example, mass amounts of consumer data can be aggregated and clustered to gain insights into customer behaviors and trends. City planners can use clustering to identify groups of housing based on house type, value, and geographical location. [reference] Property and casualty insurers can analyze variables related to staged car accidents resulting in fraudulent insurance claims. Some of these variables are low vehicle value, average number of visits to the chiropractor, and absence of a police report. The challenge this report attempts to address is distinguishing distinctive fields from a spatial image consisting of multiple spectral bands.

The Indian Pines [reference] dataset used in our research consisted of two-hundred wavelength bands, and was taken over a landscape of sixteen different field types such as alfalfa, grass, grass-mowed, corn, etc. As an example of the challenges present in satellite image processing, viewing a standard red, green, blue image of the fields from above may not give enough useful information to differentiate between fields such as grass and mowed grass. Using a hyperspectral sensor, on the other hand, can provide a vastly larger amount of data, as well as many different types of data, by collecting more wavelengths than are visible to the human eye. Certain wavelengths (such as infrared) by themselves can provide useful information, as well as unique combinations of many bands. By contrast, some wavelengths contain no useful identifying information whatsoever. The challenge lies in implementing a suitable unsupervised clustering method to parse through the spectral bands to find significant clusters representing fields.

II. Implementation

A. K-Means & PCA Implementation

In order to efficiently implement the different algorithms necessary to solve this unsupervised clustering problem, we decided to use a popular python machine learning library, scikit-learn reference.

We first decided to implement the K-Means algorithm. The problem seemed to fit nicely into a K-Means problem since we knew in advance how many data clusters (fields) existed, which is an important initialization parameter for the algorithm. We used the scikit-learn's K-Means reference implementation because it allows for a good amount of fine tuning through its many parameter options.

We next looked into dimensionality reduction. The initial dataset is 145x145 pixels in size with 220 spectral layers for each pixel. We again used scikit-learn to implement Principal Component Analysis, or PCA. This implementation also allows for whitening, which can be used as another parameter during the experiments to find the best method.

In order to gain a better understanding of the data, we developed a simple tool using python and matplotlib to visualize each layer of the spectral data. Figure 1 shows the simple interface. After selecting an X, Y spatial position and a spectral layer number, the top graph shows the relationship between the currently selected pixels spectral data (in blue) and the previously visited pixels data (in red). The bottom-right graph shows the visual representation of the specified spectral layer. This allows us to easily see each layers impact on the final clustering of the data. The bottom-left graph shows the final ground truth of the dataset for reference, along with a black point indicating the current X, Y position.

Using this tool, we were able to create a normalization function to weight the more distinctive spectral layers more heavily, as shown in Figure 2, while assigning a smaller weight to the less indicative layers such as shown in Figure 3.

After applying normalization on the spectral layers, we experimented with giving a pixels spatial coordinates greater impact in the final clusterings. Because the initial data processing converted the data cube into a set of feature vectors, spatial information was lost. To fix this, we append the X and Y position onto the spectral feature vector and

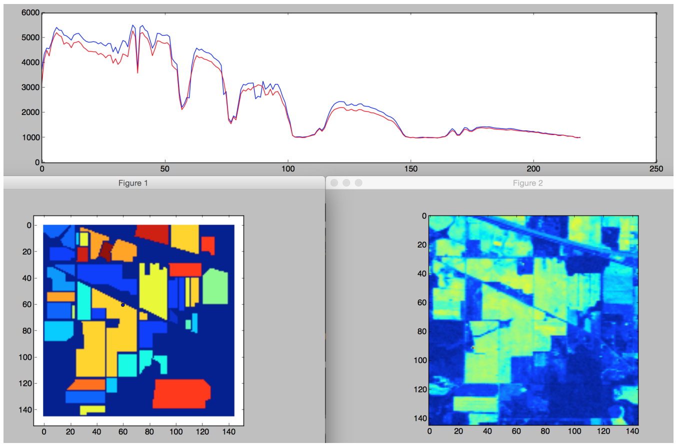

Fig. 1: Spectral Data Viewer

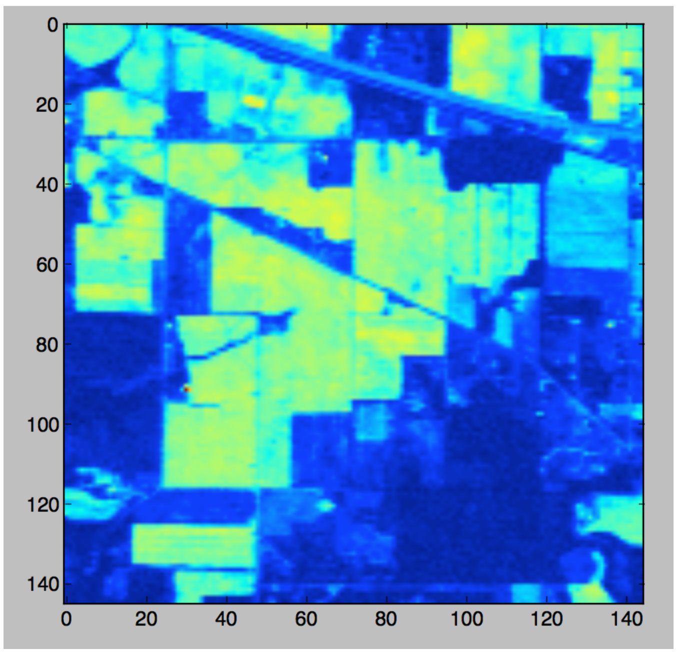

Fig. 2: Good Spectral Layer

factored the X and Y weights into the normalization.

B. Neighborhood implementation

Another look at the data revealed that neighboring pixels are very often in the same class; it seems intuitive that spatially close areas have similar spectral signatures. This observation can be used to enforce a bias on any proposed clustering provided by K-means or any other clustering algorithm. The neighborhood biasing algorithm we created iterates through each pixel, surveying its neighbors within a defined spatial radius, and finds the mode of the classes found. If a certain class is found to be very predominant in the immediate spatial neighborhood, the class of the current pixel is changed to that class. The algorithm is iterative and can be run as many times as desired until convergence. The result of the neighborhood bias algorithm is the removal of

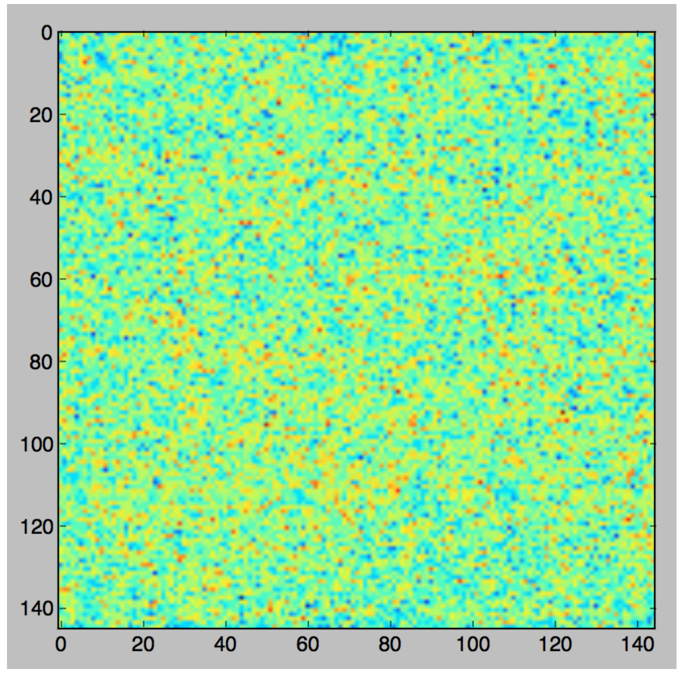

Fig. 3: Bad Spectral Layer

spectral outliers in otherwise well-defined clusters and an overall smoothing of the output provided by some clustering algorithms. This approach can be beneficial if there are many spectrally noisy outliers, but could also be bad if there are small clusters that could be smoothed over by the algorithm. The radius of the polling neighborhood and the number iterations can control the level of smoothing. In our case, we generally used a small radius of 2 pixels and 10 iterations of the algorithm.

III. Experiments

In order to test for the best approach, we used the base implementation of K-Means with 16 cluster centers as a baseline for the following experiments. We also used both the rand index and the adjusted rand index as a quantitative measurement of accuracy. We used the adjusted rand index as a reference to have a clearing result since as the data set increases so does the randomness causing the rand index to be higher than it should be.

From the implementation described above, there are many parameters ranging from the normalization weighting, to the use of PCA and whitening that all need to be set in order to get an accurate clustering result. In order to have a homogeneous computing environment that we could set for long running jobs, we set up a server using Amazon Web Services [REFERENCE]. We then ran a script to run the algorithms with the different parameter sets. Figure **??** show the different parameters that were tested and their values.

Table of sam's bullshit

From these experiments we were able to see the different trends in the parameter sets. The first trend was in the

PCA	Whitening	Spectral Weight	Band 167 Weight	Spatial Weight	Num Cluster
TRUE	TRUE	0.01	1000	1	16
TRUE	FALSE	0.001	2000	2	17
		0.002	4000	4	
				10	

Fig. 4: Parameters for Experiments

normalization set. The x and y position had a maximum effect at a weight of 2 times the original. The overall spectral weighting was best at .001 times to make it comparable to the positional. The experiment also shows that the spectral layers 167, 52, and 120 give the best impact on the final clusters weighted at 2000, 100, 100 respectively. We also found that using PCA had a negative effect on the final clustering result since the PCA would form the spectral clusters together based off their variance not saving the spatial representation.

This set of parameters produced a rand index of 0.883907 and a adjusted rand index of 0.095498 which is a great improvement on the baseline K-Means algorithm of 0.820231 for the rand index and for 0.034969 the adjusted rand index.

In order to test the neighborhood bias algorithm, we simply used the rand index and adjusted rand index to compare with out baseline and other algorithms. The algorithm came up with .886272 for the rand and 0.100829 for the adjusted rand index which is also significantly better than the baseline of 0.820231and 0.034969 respectively.

IV. Conclusion

After implementing our SVM and optimizing parameters, we were able to consistently achieve classification rates between 65% and 74% on randomly chosen test data. In the process of developing this SVM we found that we often wished we could change global parameters like C and the variance of the radial basis function depending on the class of the data; we found that some letters like k were misclassified much more often than others. After all of the experiments we found that having parameters C = 4, 1 for the sigma of the RBF, 75% of the total dataset and 3 bootstraps. In the end we would have liked to implement a tree style approach that would better classify closely related letters such as i and j by having a SVM trained on the two letters specifically instead of just 1 vs the rest.

References

[1] Bishop, Christopher M. *Pattern Recognition And Machine Learning.* New York: Springer, 2006. Print.

[2] Zhang, Hantao. *Unsupervised Learning.* http://homepage.cs.uiowa.edu/h̃zhang/c145/notes/chap18b.pdf. N.p., 2016. Web. 4 May 2016.

[3] Francis, Louise. *Applications of Unsupervised Learning in Property and Casualty Insurance with Emphasis on Fraud Analysis* http://www.slideshare.net/salfordsystems/final-applications-of-unsupervised-learning-in-pc. N.p., 2012. Web. 4 May 2016.

[4] Buitinck et al., *API design for machine learning software: experiences from the scikit-learn project.* ECML PKDD Workshop, pp. 108-122, 2013.

[5] Matteo, Matteucci. *A Tutorial on Clustering Algorithms.* http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/. N.p., 2016. Web. 4 May 2016.

[6] Purdue Research Foundation. *Hyperspectral Remote Sensing Scenes.* http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes. N.p., 2016. Web. 4 May 2016.

[7] Amazon. *Amazon Web Services.* https://aws.amazon.com/. N.p., 2016. Web. 4 May 2016.

[8] Blondel, Mathieu. *Fuzzy K-means and K-medians.* https://gist.github.com/mblondel/1451300. N.p., 2011. Web. 4 May 2016.