CS221
Final project report

Matthias Pall Gissurarson
mpgis@stanford.edu

Vilhjalmur Vilhjalmsson
villi@stanford.edu

ABSTRACT

Working with data from the Magellan expedition, we built a classifier to classify whether a patch of an image contained a volcano or not. Using a machine learning pipeline comprised of normalization, principal component analysis and a support vector machine with a radial basis function, we built a classifier that achieved a high ROC AUC score, higher than a baseline classifier provided with the data.

1. INTRODUCTION

The data we're working with are images that come from the radar on the Magellan expedition to Venus [2]. The radar beam was projected from the satellite down onto the surface, and the intensity of the back-scatter then recorded. These images have then been pre-processed by JPL, using computer vision techniques ("focus of attention") to detect possible areas that might contain volcanoes. Examples of images from the data set can be found in figures 1, 2 and 3. These areas were then labeled by geology experts with a label of 0 to 4, 0 indicating that there was not a volcano in the picture, 1 being "definitely a volcano", 2 "probably a volcano", 3 "possibly a volcano" and 4 "only a pit is visible". The data from the area is then aggregated, so that 4 pixels become one in the data set called "chips". These chips contain a list of 15x15 floating point numbers representing matrices, each of which indicates an area. They come with the aforementioned labels, and are split into different data sets, and each data set into a test set and a training set. We used only two of the available data sets, C1 and D4.

Description of the work is outlined in "Learning to Recognize Volcanoes on Venus (1998)" by M. C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, L. Crumpler and J. Aubele [1]. Burl et. Al's paper also defined a classifier, which we implement to compare ourselves against.

1.1 Justification

The goal of this project is to identify whether a piece of an image contains a volcano or not. We have very few positive samples, so judging models by misclassification error would be useless, as getting a 3% error would be trivial by always guessing "not volcano". It is intended to ease researchers job and spare them the pain of going through thousands of images, and reduce the set down to a manageable size. We thus want a high true positive rate, but a low false positive rate, i.e. trying to get a good TPR/FPR ratio while still finding a significant portion of the volcanoes. This can be measured using area under the curve in a ROC plot.

The justification is that researchers do not have the time to manually label and identify volcanoes by hand, and a computer assisted method would really help them with their research.

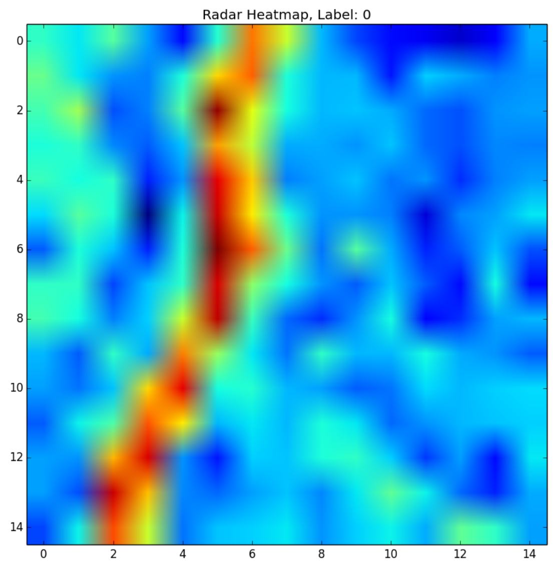

Figure 1: A picture from the data set with the label 0 (Not a volcano)

2. HYPOTHESIS

Burl et. Al define a bayesian classifier in their report, which is intended as a baseline for evaluating how well other classifiers do on the data. Our hypothesis is that by using features learned by clustering, we can do better than the

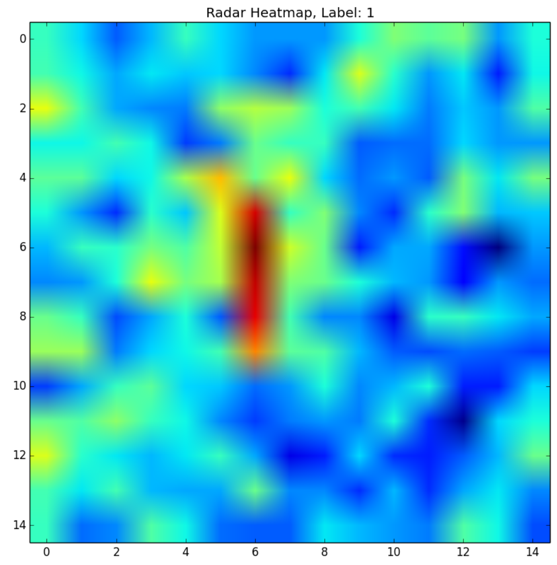

Figure 2: Labeled 1: (Definitely a volcano)

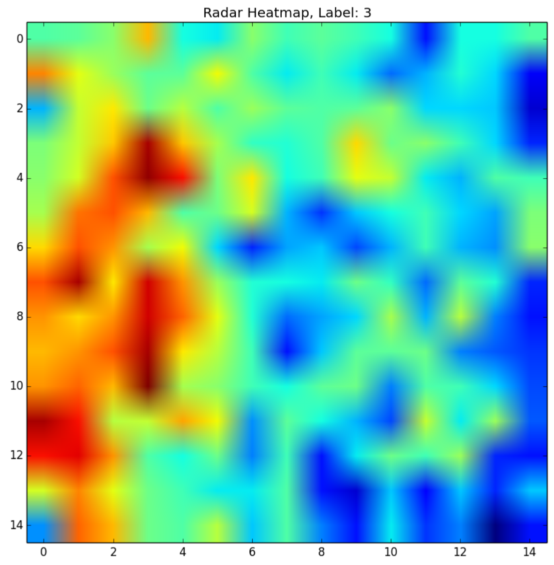

Figure 4: Labeled 3: (Possibly a volcano)

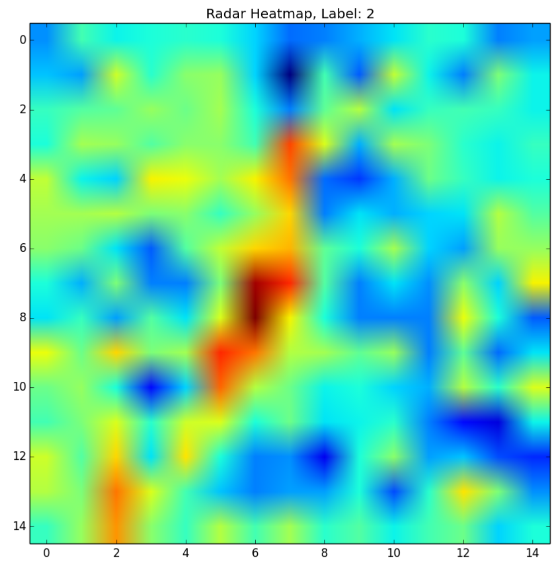

Figure 3: Labeled 2: (Probably a volcano)

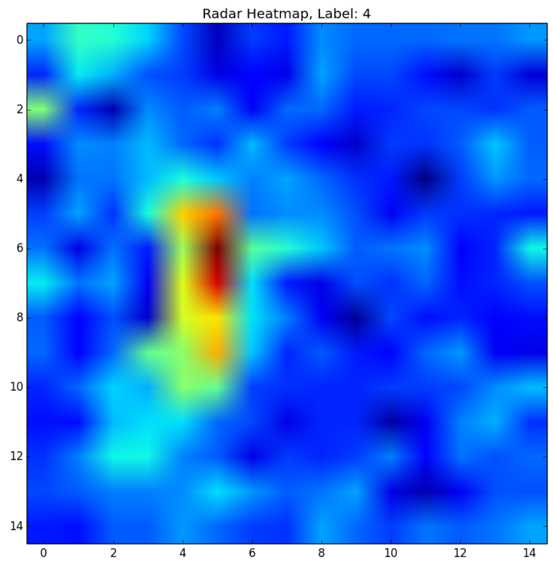

Figure 5: Labeled 4: (Only a pit visible)

baseline classifier. As the best metric in our case is the area under curve on a ROC plot, which gives an estimate on how well a classifier is doing via the fpr and tpr classification rates. The Bayesian baseline classifier has an AUC of 0.85, and our hypothesis is that we can beat this.

3. PROBLEM MODEL

3.1 Selection

The features of each image that are possibly interesting have been found by the "focus of attention" algorithm as applied by JPL already. Thus, we decided to use the entire matrix for each data point in each chip. Every piece might contain the telltale signs of a volcano, or reveal that it is actually just a canyon. We could possibly have dropped all but the middle of the matrix, as the computer vision algorithm centered on the feature it deemed volcano like, but again, this might drop data that would imply a canyon.

3.2 Preprocessing

As the chips are from various sites on Venus, each taken at a different time and angle, the intensity values did not mean the same in all the pictures. It might be from ground that has different texture, maybe taken during the different time of day, or other confounding factors. Thus, we normalize each matrix so that each of its values represents how many σ's the value was from the mean of the matrix.

3.3 Transformation

After the selection and pre-processing, we had matrices that were 15x15 and normalized. This means that it had 225 features, one for each value of the matrix. Considering how unbalanced the labels are and how few positive samples are in the data set we knew that this would be way to many features to be able to meaningfully use for data mining, so we used a feature reduction method called principal component analysis, or PCA for short. What principal components are is basically vectors that best represent the data in a given number of features. These principle components are found, and then each data point is transformed into the lower dimensional space represented by the principal components. Using this we could reduce our features from 225 to any number we want. With decision trees, a value of 5 worked very well, but using SVMs, values from 40-80 worked much better.

3.4 Machine Learning

3.4.1 *The baseline classifier*

The first classifier we tried was a Bayesian classifier used in the original testing of the data, in Burl et. Al's paper, which will be used as a baseline. The idea is to find the means of the positive and negative classes and assume the class-conditional densities are Gaussian. Then we can calculate the probability of seeing a image given it is of a particular label, $p(\omega_i|y_i)$, where ω_i is an image and y_i is its label. From Bayes theorem we get

$$p(\omega_i|y_i) = \frac{p(y_i|\omega_i)p(\omega_i)}{p(y_1|\omega_1)p(\omega_1) + p(y_2|\omega_2)p(\omega_2)}$$

where $p(y_i|\omega_i) = N(y_i; \mu_i, \Sigma_i)$ and can be calculated using a probability density function. This approach worked reasonably well and resulted in an AUC of 0.85, which will be our baseline for measuring other classifiers.

3.4.2 *Initial experimentation*

We started out by implementing AdaBoost in python. This AdaBoost used weighted random sampling with replacement to weigh the examples by the weights, so that it did not need a classifier that takes in weights. We then implemented K-Means clustering, to find centroids in the data, which, when applied to images, resemble edges in the images. We then extracted features for each data point by calculating by how much the data point resembled these edges. These features were then run through a linear gradient descent classifier, which used a logit function as a hypothesis function. This approach was very slow, and did not yield good results at all. The slowness also hindered much testing, as every run took a long time.

We decided that K-Means was not the way to go for feature reduction, and decided to use PCA, principal component analysis, to determine features. Principle component analysis aims to find vectors which best describe the data if you take a linear combination of them.

We initially tried to use these as data in the classifier, but the PCA and K-Means seemed to clash a lot, yielding poor results. We then stopped using the K-Means, and it got a little better, but not as good as we hoped.

3.4.3 *AdaBoosted decision trees*

We then tried using decision trees. These were also implemented in python by us, but were perhaps very suboptimal in their implementation, as they were slow an unpruned. They used entropy as a measure of how good the split was. It was with them that we realized that the classifier was going to need a lot more examples of volcanoes so that it could do its job, so instead of using just the initial ordering of the data set, we transformed it such that all the volcanoes that we were sure of were in the list, and then some variable amount of non volcanic examples, as determined by the ratio of volcanoes to non volcanoes.

It was at this stage that we had finally implemented plots, and we could finally begin to classify the models based on AUC. The first AdaBoosted model did not work very well, yielding a AUC of 0.55. We decided that maybe it would be better to have the ratio more precise, so instead of running the whole data set through, we selected a random subset with replacement for every run.

We then fiddled around with the parameters a lot, varying the ratio of volcanoes to non volcanoes, and the amount of features PCA reduced the set to. We found that a value of 5 PCA features were good, and that might be because of luck in the AdaBoost random selection from the weighted dictionary, but we got a plot that was pretty impressive for only 5 features per data point, as can be seen from it's ROC curve in figure 9.

3.4.4 *Other classifiers*

After trying for a long time to find good values for the AdaBoosted decision tree, we could never get it's AUC to go above 0.65. We then turned to other classifiers, to see if we could beat the AdaBoosted trees. Without selecting any special subset of the data in particular, we used each classifier directly on the PCA transformed data, with various settings. The results of these trials can be seen in 1.

We tried some classifiers from scikit_learn, and we saw that Gradient Boosting and Support Vector Machines did a really good job, beating the classifier described in the paper. By tuning the SVM, such as using a radial basis function kernel ("rbf") it had a AUC of 0.90.

We fiddled with other types of SVM and they did much worse, thus resulting in using the SVM with the rbf kernel as our final model.

As a late addition, we tried to send the normalized features through an artificial neural network to learn features, and then piping that to an svm for classification. It preformed very poorly, essentially not doing anything. We also tried to pipe those features to a logistic regression classifier, and it did not do well either.

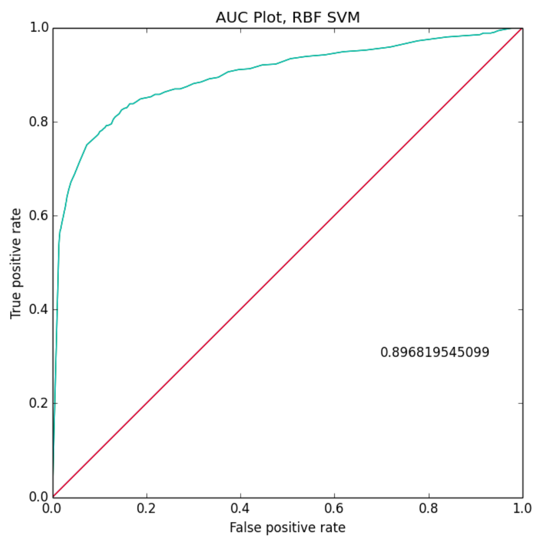

Figure 6: ROC plot from a SVM using a radial basis function kernel, with 60 principal components

4. INTERPETATION/EVALUATION

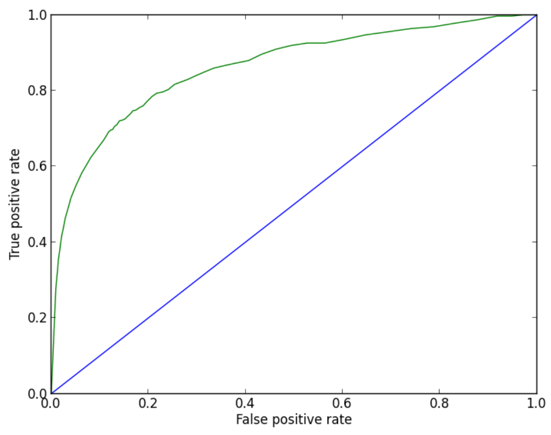

Figure 7: ROC of the Gradient Boosting classifier

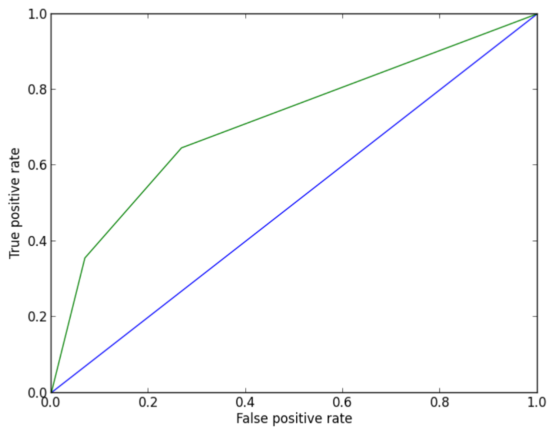

Figure 8: ROC of the Random forest classifier

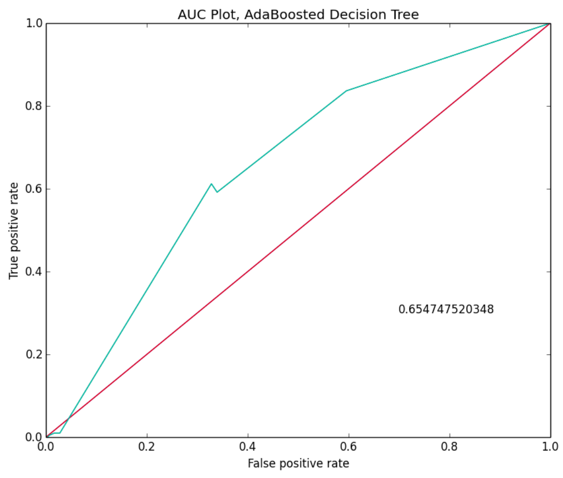

Figure 9: ROC of the AdaBoosted Decision Tree classifier

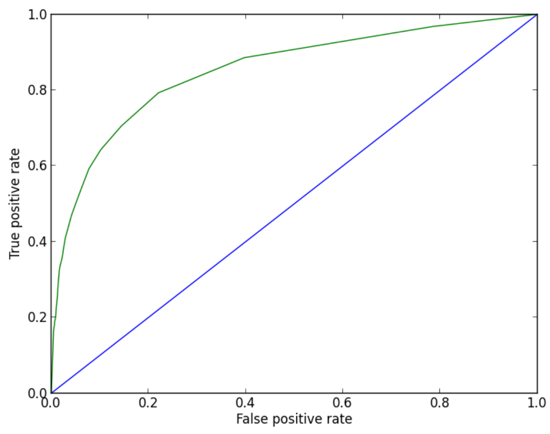

Figure 10: ROC of the Burl et. Al's classifier

Name of classifier	Transformation	Settings	AUC
AdaBoosted Decision Tree	PCA	PC = 5, Hypotheses = 5, Volcanoes to Non-volcanoes = 0.2	0.65
AdaBoosted Decision Tree	PCA	PC = 5, Hypotheses = 5, Volcanoes to Non-volcanoes = 0.1	0.49
AdaBoosted Decision Tree	PCA	PC = 50, Hypotheses = 5, Volcanoes to Non-volcanoes = 0.2	0.61
Gradient Boosted Decision Tree	PCA	PC = 40, Default settings	0.86
Support Vector Machine	PCA	PC = 5, Default settings, kernel = "rbf" ("radial basis function")	0.65
Support Vector Machine	PCA	PC = 40, Default settings, kernel = "rbf" ("radial basis function")	0.89
Support Vector Machine	PCA	PC = 60, Default settings, kernel = "rbf" ("radial basis function")	0.90
Support Vector Machine	PCA	PC = 80, Default settings, kernel = "rbf" ("radial basis function")	0.89
Support Vector Machine	PCA	PC = 60, Default settings, kernel = "linear"	0.55
Support Vector Machine	Artificial Neural Network	Default settings, kernel = "linear"	0.50
Support Vector Machine	Artificial Neural Network	Default settings, kernel = "poly"	0.50
Support Vector Machine	Artificial Neural Network	Default settings, kernel = "rbf"	0.50
Logistic Regression	Artificial Neural Network	Default settings	0.50
Random Forest	PCA	PC = 40, Default scikit settings	0.71
Burl et. Al's Simple classifier	PCA	PC = 40	0.84
Burl et. Al's Simple classifier	PCA	PC = 5	0.74

Table 1: Area Under Curve for various classifiers and settings

The AUC value of the various models we tried can be seen in 1. The ROC plot for the RBF SVM can be seen in figure 6, and the others can be seen in figures 7,8,9 and 10. These values were found by running each model a few times, using different splits of our data set. We evaluated the models by taking the entire data set and randomly shuffling it, and then split it into 5 parts. We then used 1 part as a test set, and trained the model with the other 4 parts. We then did this for all the parts and taking the average AUC score of the results, thus preforming cross evaluation.

After much trial, we were very happy with how well the support vector machine turned out. It seems that using the "rbf" kernel is the trick, though we can't really say that we understand why it works so well, as we do not have enough experience with kernels, and SVM are not very opaque when using complicated kernels.

5. CONCLUSION

As we can see, support vector machines out preformed all of the other models, and reaching an AUC of 0.90, which most consider to be quite good. We beat the AUC of the Bayesian classifier, showing that using an SVM out preformed a Bayesian assumption. Our K means clustering feature learning method did not work at all.

We though that artificial neural networks would be good at this area, but it seems that the features are very confusing. We had a look at some of the pictures ourselves, and could often not deduce anything from them, leading us to think that this is probably not a project suited for the kind of image recognition artificial neural networks use.

Next steps might be to use even more advanced feature selection techniques, and try to get even more examples classified by the experts to use for training. This could be an online process, with a classifier that learns from each example it presents to experts as a possible volcano, and dynamically updates itself as it learns more. We could not test this, as we have no access to experts, but it could improve classifiers quite a lot.

6. REFERENCES

[1] Michael C. Burl, Lars Asker, Padhraic Smyth, Usama Fayyad, Pietro Perona, Larry Crumpler, and Jayne Aubele. Learning to recognize volcanoes on venus. *Machine Learning*, 30(2-3):165–194, 1998.

[2] Michael C. Burl, Lars Asker, Padhraic Smyth, Usama Fayyad, Pietro Perona, Larry Crumpler, and Jayne Aubele. Volcanoes on venus - jartool experiments, 1999. [Online; accessed 17-August-2013].