Wildfire Burn Area Prediction

SUNet ID: eklyce, csun3, rysotomo
Name: Eliza Klyce, Cuthbert Sun, Rysen Otomo

Abstract

Our project seeks to predict the burn area of wildfires, based on certain starting factors. Intuitively, these factors include features such as temperature, wind speed, humidity, etc. Combining these features with more advanced metrics (FFMC, DMC, etc.), we used various machine learning techniques (Discriminant Analysis, SVM, neural networks, etc.) to try and predict the wildfire size. In the end, our algorithms sought to classify fires into 5, well-defined size-classes.

Introduction

Recently, the northern and southern California fires have destroyed large portions of land and affected the lives of many people. Being able to predict the eventual size of these fires could have a significant impact on evacuation, containment, and fire-fighting efforts. Our project began as a an attempt to predict the likelihood of wildfires occurring in specific areas. However, predicting the locations of future fire occurrences proved rather difficult - largely because of data shortcomings. As we continued to research wildfires, we refocused our efforts on attempting to predict the size (in Ha) of a particular fire, based on factors from the day the fire started. Initially, we implemented a regression to predict the raw burn area, but this proved heavily skewed because of our data set. As a result, we focused instead on classifying the size of a wildfire into five different categories. This change to predict size rather than location was inspired partially by the greater availability of data on the subject and also by a desire to better understand the factors that contribute to the spread and intensity of fires. We hope that a greater understanding of these factors will help to better identify massive fires in critical areas and thus improve containment efforts. Further, we hope that a better understanding of how and why fires spread will lead to a better understanding of how they start in the first place.

Related Work

The most relevant prior work comes from UC Irvine by Paulo Cortez and Anbal Morais. One approach that they had was to use $y = \ln(1 + x)$ when dealing with 0's in the data. Quite a few forest fires, of the 517 reported, were said to be of size 0. Our approach to this issue was simply to cut out all instances of fires of size 0. This significantly reduced our sample size and also contributed to the imbalance in our buckets. Their approach allowed them to include all 0's and was very clever. One similarity between their work and ours is that they had the most success with an SVM algorithm. Their algorithm was however slightly more accurate probably due to their treatment of the 0's. They also compared different algorithms. Another paper by V. Cherkassy and Y. Ma gave us the idea to add noise to sparse labels. We decided that like them, it could be good to add Gaussian noise. They also used three types of non-Gaussian noise.

Data Set and Features

Our data set comes from the Montesinho natural park in Portugal and was collected from January 2000 to December 2003. This is the same data set that Paulo Cortez and Anbal Morais at UC Irvine used. Each time a wildfire occured, data was collected. Of the features collected, the ones used in our project were: the **FFMC** (Fine Fuel Mosture Code), the **DMC** (Duff Moisture Code), the **DC** (Drought Code), the **ISI** (Intraseasonal to Interannual), the **temperature** (in °C), the **relative humidity** (as a percent), the **windspeed** (in km/h), and finally the **outside rain** (in mm/m^2). In total, we had 8 features.

PCA

Although there were 8 features included in our dataset, we determined that not all of these features were very meaningful for either regression or classification. In order to visualize the relationship between the features and the area of the wildfire, each feature for a data point was plotted against the output. The plot is shown on the next page. Looking at each plot, it is clear that there is not a strong correlation between the features and the output space. Granted, there are several apparent outliers in fire size, but even with them removed the data points remain quite scattered.

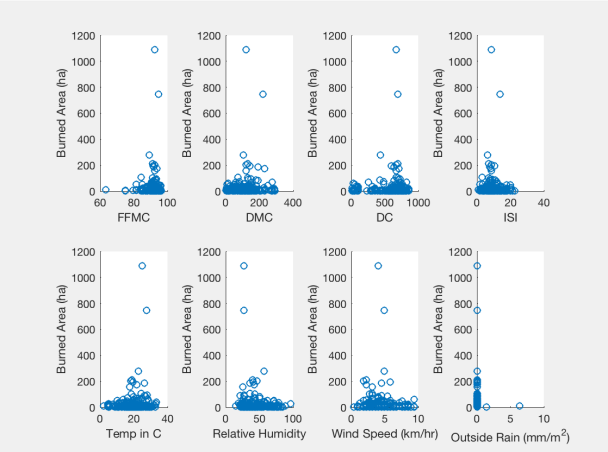

Figure 1: Output vs. Features

Knowing this, PCA was used to determine which features contributed the most to the variance of the output space. The chart below describes the effect of each feature on output variance.

Feature	Percent of Total Variance
FFMC (Fine Fuel Moisture Code)	85.0670
DMC (Duff Moisture Code)	11.3705
DC (Drought Code)	3.0889
ISI (Intraseasonal to Interannual)	0.3601
Temperature (°C)	0.0702
Relative humidity (percentage)	0.0227
Wind speed (km/h)	0.0160
Outside rain (mm/m^2)	0.0044

From PCA, the first four features (FFMC, DMC, DC, and ISI) were responsible for over 99% of variance. Thus, it was decided that we would reduce the input space to only these features for non neural network tests.

Classification Problem

Initially, our goal was to create a regression, and predict directly the burn area of the fires. This produced mixed results - at best our Neural Network had a 30 Ha RMSE. When you consider that our fires range from < 1 Ha to > 1000 Ha, this doesn't seem so bad. Unfortunately, this performance is largely masked by the fact that $\approx 90\%$ of our data is less than 30 Ha, this means we have at least 100% error for those 90% of the data set. As per advice received on our milestone, we chose to split our outputs into 5 buckets. According to the National Wildfire Coordinating Group (nwcg.gov), the bucket dividers based on different class size fires are [0 0.25 10 100 300 1000] acres. This allows us to change our regression problem into a classification problem.

Methods & Algorithms

1. Naive Bayes equation:
   $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$
2. SVM Kernel:
   $K(x, x^{'}) = exp(-\lambda||x - x^{'}||^2), \lambda > 0$
3. Neural Net: To implement a neural net, we used Matlab's Neural Network Toolbox. For the classification problem, we have one hidden layer (sigmoid activation layer), and a softmax output. Our cost function was the cross entropy between the predicted classes and the target classes. Lastly, the training function was the Scaled conjugate gradient backpropagation. All of the training functions performed similarly. The main design knob we had was the size of the hidden layer. It turns out that 10 hidden neurons is roughly optimal - too many hidden neurons and the network has a tendency to overtrain on the training set and perform poorly on the test set. We didn't apply PCA to the neural net because neural nets learn how to weight the inputs, so the irrelevant features will be z̈eroedöut. Further, relying on that fact, we tried to create a higher-order input features set, by matrix multiplying the 8 inputs to produce 64 second order inputs (some redundant). Adding these to the original 8, we tried running our algorithm with 72 higher-order inputs.

Results & Discussion

Initial Results

With the original data set, we achieved the following accuracies:

Algorithm	Percent Misclassification (LOO)
Naive Bayes	51.48
SVM with Gaussian Kernel	49.63
Neural Network	49.94

New results with added data

Algorithm	Percent Misclassification (LOO)
Naive Bayes	61.11
SVM with Gaussian Kernel	50.00
Neural Network	48.66

Addition of Gaussian Noise

Because of imbalances in our data we decided to add Gaussian noise to our models. We added Gaussian noise to categories with fewer fires, namely categories 1, 4, and 5. The distribution's changed as follows:

Classification	Original Num Samples	New Num Samples
1 (0 - 0.25 acres)	1	**51**
2 (0.25 - 10 acres)	107	107
3 (10 - 100 acres)	131	131
4 (100 - 300 acres)	22	**52**
5 (300+ acres)	7	**57**

The noise was based on random samples from each category —except in the case for classification 1 where there was only one sample.

While we did not see an improvement in percent misclassification, our algorithms did have a much more evenly dispersed guesses over all. Similarly, we can see from this matrix that most misclassifications were off by only one bucket. Take for example, the confusion matrix of our Gaussian Kernel SVM model with noise versus without, which was one of our most successful.

Confusion Matrix for Gaussian Kernel SVM model with noise:

		Predicted				
		1	2	3	4	5
Actual	**1**	0	1	0	0	0
	2	4	25	73	2	3
	3	0	31	95	3	2
	4	0	3	11	7	1
	5	0	1	3	0	5

Confusion Matrix for Gaussian Kernel SVM model without noise:

		Predicted				
		1	2	3	4	5
Actual	**1**	0	1	0	0	0
	2	0	29	75	2	1
	3	0	31	99	1	0
	4	0	3	14	5	0
	5	0	1	4	1	3

Discussion of Neural Network Performance

Real Data Results

Below are the figures for the confusion matrix and the ROC. We see that the neural network almost always guesses either class 2 or 3. In large part, this is because the data set is composed of mostly class 2 or 3 data points (almost 90%). Thus, the algorithm learns to only guess 2 or 3. The ROC curve looks fairly good, though, with the guesses for all of the classes (except class 1) very close to the ideal line. Unfortunately, if we average the misclassification number over 20 training and test iterations (to reduce the effect of random chance, given our small data set), we get a 49.94% error rate, which isn't great.

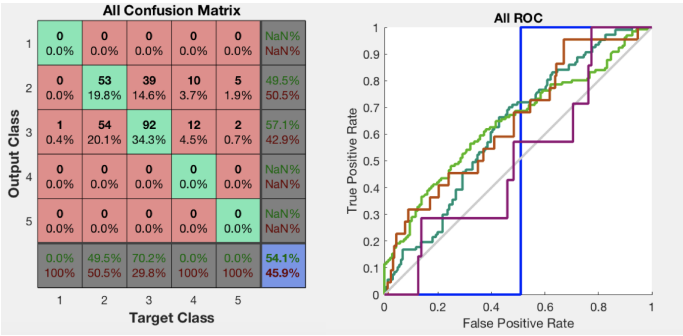

Figure 2: Confusion Matrix and ROC with true data

Fake Data Results

If we include Gaussian Noise such that classes 1, 4, and 5 have 50 data points, we get marginal improvement in the accuracy, to 48.66 % (again, averaged over 20 training/testing iterations). While this isn't much of an improvement, the confusion matrix gives us a marked improvement. Now, the neural network has much

more spread output classes, and is very accurate on its guesses for Class 1 and Class 5. This is likely because there was a large number of fakedata points created for those classes. Another promising aspect is that the performance for classes 2 and 3 is maintained. This is especially important since those were the real data points. Upon further inspection, we see that there is a lot of cross classification. This actually isn't a huge deal because they are neighboring classes, so the accuracy in distinguishing between the two isn't critical.

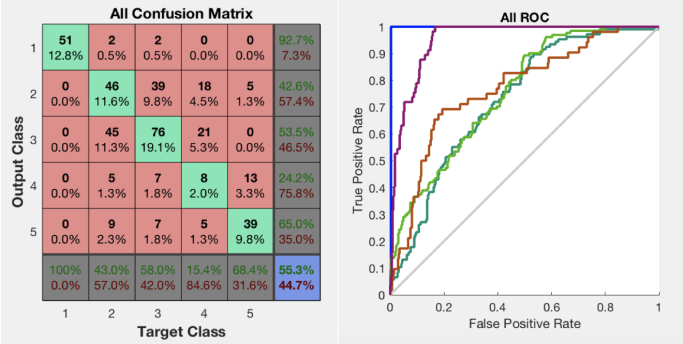

Figure 3: Confusion Matrix and ROC with added data

Conclusion and Future Work

We see that the performances of all the varying algorithms was all pretty similar. As discussed, this is mostly a product of our data set being heavily populated with data points from the same two classes. In an attempt to alleviate this effect, we added Gaussian White Noise to the other classes to add fakedata points. This worked to some degree, but the sheer number of fake data points is still questionable.

Because the shortcoming of our learning algorithms was in the data set, the immediate future work should be focused on collecting data in that respect. The 8 features seemed to work pretty well (on fake data, the algorithms performed slightly better), so we really just need more data, and data evenly distributed through the 5 fire classes.

It should also be noted that misclassification isn't really as binary as it is reported. Misclassification, when guessing a neighboring class, isn't nearly as egregious as when a class 1 fire is predicted as a class 5 or vice versa. In this regard, our algorithm actually performed better than reported.

Acknowledgement

We would like to gratefully acknowledge the CS229 teaching staff for providing all the help, insight, and advice in making this project a great learning experience. The suggestions helped us determine a better approach to meet the goals of this project.

References

(1) Cortez, P., Morais, A., A Data Mining Approach to Predict Forest Firesusing Meteorological Data
(2) B. Arrue, A. Ollero, and J. Matinez de Dios. An Intelligent System for False Alarm Reduction in Infrared Forest-Fire Detection. IEEE Intelligent Systems, 15(3):6473, 2000.
(3) V. Cherkassy and Y. Ma., Practical Selection of SVM Parameters and Noise Estimation for SVM Regression. Neural Networks, 17(1):113126, 2004.
(4) Safi, Y., Bouroumi A., Prediction of Forest Fires Using Artificial Neural Networks, Applied Mathematical Sciences, Vol. 7, 2013, no. 6, 271 - 286
(5) Communications System Toolbox - MATLAB
(6) Statistics and Machine Learning Toolbox - MATLAB