Probabilistic Dust Storm Predictive Modeling

Sean Flaherty

July 28, 2017

Acknowledgements

NMSU Department of Entomology, Plant Pathology and

Max Bleiweiss

NMSU Department of Plant and Environmental Sciences

Merrill Bean

Dr. Coly Brungard

Dr. David DuBois

NMSU Department of Computer Science

Dr. Huiping Cao

NMSU Department of Mechanical and Aerospace Engineering

Josue Gutierrez

Contents

List of Figures

1 Introduction

Dust storms have numerous adverse effects to populated areas, including disruptions to communications and traffic, damage to property and equipment, and respiratory health hazards. These effects are caused by reduced visibility and dust particles suspended in the air. Further effects, such as damage to croplands, are caused by sediment that settles after dust storms. To mitigate the damages caused by dust storms, an early warning system is ncessary. To develop an early warning system, a predictive model is needed. Currently, there exists a numerical model known as the Dust Regional Atmospheric Model, or DREAM. DREAM models dust particle movement such that it can predict a dust event up to 72 hours in advance, and attempt to predict the movement of dust particle systems [1].

The Public Health Applications in Remote Sensing (PHaiRS) project aimed to assimilate higher-resolution data into a version of the DREAM model adapted for the Southwestern United States referred to as DREAM/SW with the goal of increasing the prediction accuracy of DREAM/SW and improving public respiratory health outcomes. While parameters of increased resolution were in agreement for overall dust movement patterns, increasing the resolution of parameter data had a plateau effect in accuracy and public health outcome [2]. Because DREAM is a deterministic predictive model, in order to determine what variables may be stronger predictors for future dust events, a probabilistic model is necessary.

1.1 Research Overview

A previous study on probabilistic modeling of dust events uses 500mB geopotential height patterns as a univariate predictor of dust events using grayscale image processing. In this study, 500mB geopotential heights are recorded in the southwestern United States and northern Mexico from 2011 to 2014 and dates are classified as dust or non-dust days. To create a more separable data set, days before and after dust days are omitted from the data. The training data for the study is formatted into a grayscale raster image, and the images for dust and non-dust days are averaged together. To perform the classification, a sample raster image is compared to the average dust day image using zero mean normalized cross-correlation (ZNCC). This technique determined that dust storms occurred as a result of a low pressure system in northern New Mexico often called the "Albuquerque low," although it only worked as a predictor of more major dust events, and could not predict smaller or more short-lived events [3].

Due to the limitations of only using 500mB geopotential heighs as a predictor for dust events, a model is needed that not only predicts large-scale dust events from a major low-pressure system, but one that also predicts small and short-lived events. The objective of this project is to create a probabilistic predictive model for dust events that achieves this goal using multiple variables as predictors. Using rapid update cycle (RUC) forecast data from 2011 to 2014, a wider range of variables is available for a predictive modeling. The larger number of variables, however, introduces the curse of dimensionality [4]. In order to have a meaningful and accurate prediction, the dimensionality must be reduced such that only the strongest predictors of dust events are used. To do so, instead of using the proper training data, a normalized principal component analysis (PCA) of the data is used instead.

1.2 Problem Definition

With existing deterministic models and limited existing probabilistic models, the goal of this project is to create a preliminary predictive model for dust storm events using data mining and machine learning algorithms in order to more accurately predict meso- and micro-scale dust storm events using weather forecast data for individual locations.

2 Methodology

2.1 Data

The data used for this study is retrieved from the NOAA Operational Model Archive and Distribution System (NOMADS). The particular forecast models used to create a predictive model are the Rapid Refresh (RAP) and Rapid Update Cycle (RUC) models. NOMADS makes its data available via HTTPS or FTP. For this project, the data is downloaded via HTTPS using the GNU Wget utility. For a preliminary model, only data recorded at 18:00 GMT is used. In order to retrieve this data, the following Wget command is invoked:

```
wget -r -np -nd -A 'ruc2anl_130_*_1800_000.grb2' -e
robots=off https://nomads.ncdc.noaa.gov/data/rucanl
```

This command reads all files from each directory in the RAP/RUC parent directory, and downloads them to a single local folder [5]. This particular command gathers all .grb2 files with a 13km resolution, although 25.2km resolution is also available. The command also does not retrieve all files, as before 2007 files existed in .grb format.

The .grb and .grb2 formats are two versons of the GRIdded Binary (GRIB) format, which is the World Meteorological Organization's standard for storing observation and forecast data. The file format contains a header that describes the contents of the file, as well as binaries for each recorded weather parameter. The binaries are stored in a raster, with each point in that raster corresponding to the recorded data for a specific latitude and longitude.

In order to read a GRIB file in Python, the pygrib library is used. The library allows the files to be opened as iterators containing gribmessage objects, each of which contains the data for a weather parameter. Each gribmessage contains a parameter name, height and time information, the latitudes and longitudes for which the data is recorded, and the corresponding parameter information for each latitude and longitude [7]. Because this project is concerned with predicting dust storms on a local scale, a single index is taken from the 2-dimensional data array for each parameter.

Data is visualized using the NOAA Weather and Climate Tool. Using this program, GRIB files can be opened and and weather maps of any desired parameter can be displayed. In Figure 1, a map is generated using the WCT viewer to visualize 500mB geopotential height during a large-scale dust event. This particular event shows the "Albuquerque Low" pressure system.

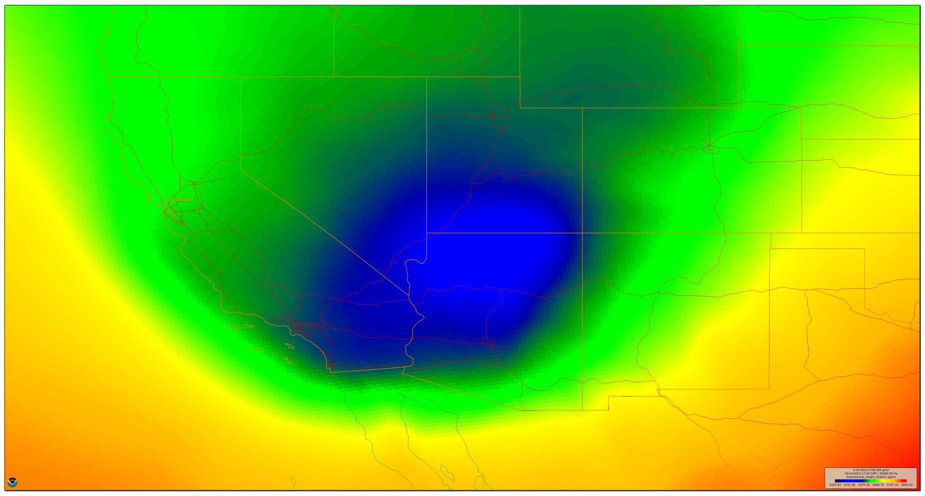


Figure 1: Map of 500mB geopotential height during a major dust event on April 14, 2012, created using NOAA WCT viewer.

2.2 Processing

In order to make a dataset readable by later machine learning algorithms, the forecast parameters for specific points from each date's raster are recorded into a CSV table containing a list of events for easier processing and lookup. An event is defined as a row of parameters corresponding to a given date and position in the GRIB raster corresponding to a latitude and longitude. An event can be classified as either a dust event or a non-dust event. If a date in the specified range contains a dust event, the location of the event at that date is used. If there is no dust event on a date, then a random latitude and longitude are randomly generated following normal distributions with the means and standard deviations of the latitudes and longitudes. The first element in every row of a CSV will be a 0 or 1, marking the event as non-dust or dust, respectively. For the raw data, the following elements in the row will be each weather parameter associated with that event. The list of dust event dates and their locations is provided by Dr. David DuBois of the NMSU Department of Plant and Environmental Science.

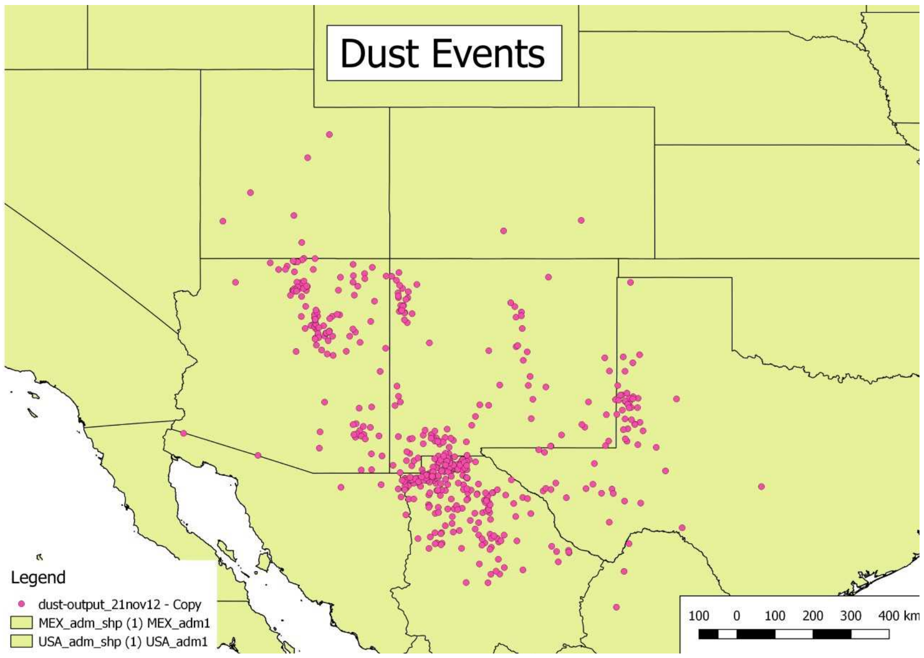
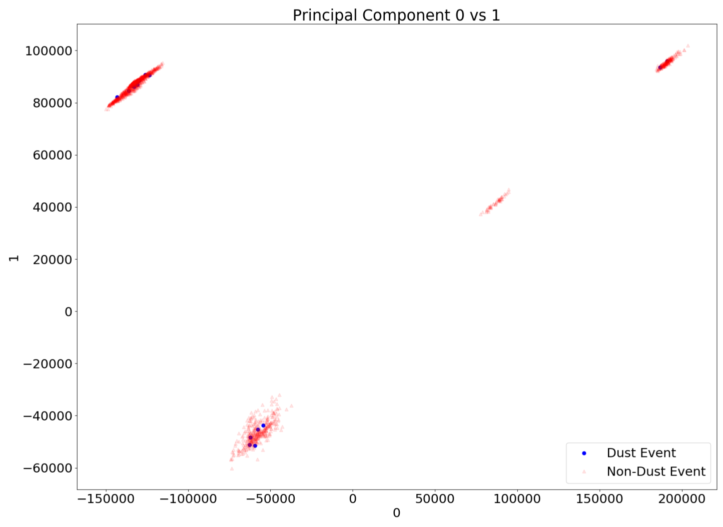
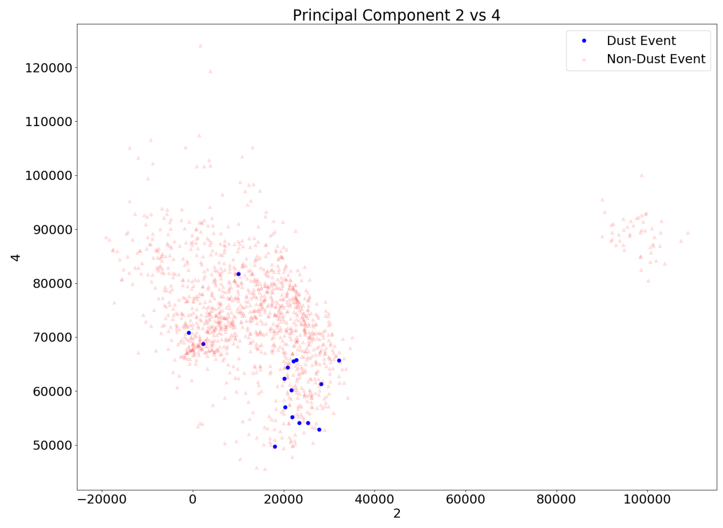


Figure 2: Map of dust events created by Josue Gutierrez using data provided by Dr. David DuBois that are used to generate the set of training data.

If an algorithm is to use normalized principal components instead of the raw data, a principal component analysis script will take the CSV files containing the weather data and write new CSV files containing the data in the form of the first k principal components, where k is specified in the script. After the CSV is written, a smaller fraction of the list of events is cut and pasted into a new CSV file, such that the larger file acts as the training data and the smaller is a set of data to test the model after training. Similar to the CSV containing raw data, each event is headed by a 0 or 1 indicating it as a non-dust or dust event.



(a)



(b)

Figure 3: Two scatter plots for principal components, one with no clear distinction between dust and nondust events (a), and one with some visible distinction (b).

The plots in Figure 3 are taken from a PCA of data from the greater Phoenix area from 2008 to 2012. The dust events in this plot are gathered from the NOAA Storm Events Database. In this example, five components are taken. These plots are useful for determining whether there is a distinct separation in the data before using it to train a neural network and make predictions.

For a preliminary predictive model, data will be taken from the dates of April 13, 2007 to April 30, 2012. This time range was selected because it is the largest set of consistent file format and number of parameters in the NOMADS RAP/RUC archive.

2.3 Modeling

To finally create a predictive model for dust events, various types of neural networks are implemented. The number of inputs can either be the raw weather data or the principal components. The output is a single node, as this problem requires a binary classification of dust or non-dust events. The number of hidden layers and nodes can vary according to which input data is used. The first model is a simple feedforward neural network, using the raw data as input. Next is another feedforward network but with prinicpal components as the input. Recurrent neural network allow for a more context-sensitive model. The recurrent neural network models follow the long short-term memory (LSTM) architecture, which allows for previous information fed to the network to be remembered or forgotten over a specific period of time [8].

Each neural network model is implemented using the TensorFlow Python library. TensorFlow creates a graph of the computation performed and automates backpropagation for optimization algorithms. Each model created for this project uses TensorFlow's built-in gradient descent optimzer [9].

2.4 Feedforward Neural Network

The first neural network used for a predictive model is a simple multilayer perceptron. The nodes in this neural network use a *tanh* activation function, because the data does not follow a normal distribution and the sigmoid does not provide a steep enough gradient for optimization [10]. With 10 principal components as inputs, networks of varying hidden layers of 25 nodes are tested to determine which provides the most accuracy. Because the dataset provided is imbalanced, the loss function is given weights such that a greater loss is calculated from a false negative than from a false positive. Because 10.5315% of the events are dust events, a weight of 1 is provided for the non-dust output node, and 9.4953, the reciprocal of that proportion, for the dust output node. The loss function used is TensorFlow's weighted cross-entropy with logits, which for any one node is:

$$E = y * -\log(\sigma(\hat{y})) * W + (1 - y) * -\log(1 - \sigma(\hat{y}))$$

where y is the target prediction, $\hat{y}$ is the prediction made by the model, σ is the sigmoid activation function $\sigma(x) = \frac{1}{1-e^{(}-x)}$, and W is the weight associated with that output node.

2.5 Recurrent Neural Network with LSTM

The second type of neural network used as a predictive model is a recurrent neural network with a basic LSTM structure. Each of the neural network elements inside of the LSTM cell use *tanh* activation functions, similar to the feedforward neural network. Below is a diagram of the basic LSTM cell used in this model.

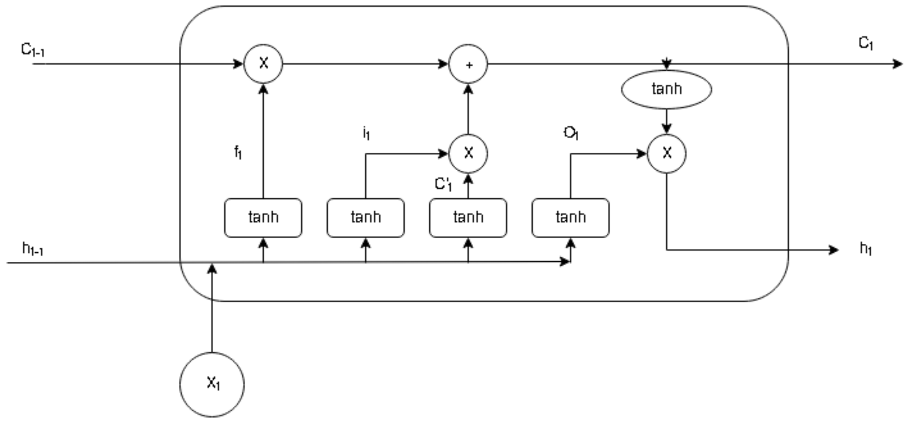


Figure 4: Basic LSTM cell, with the exception of *tanh* neural network layers instead of the usual sigmoid.

The LSTM cell takes three vectors as inputs: h_{t-1}, the previous prediction, C_{t-1}, the previous cell state, and X_t, the current input data. Because this model is a binary classifier, the size of the prediction layer is 2. The cell state's dimensions mask the prediction layer. The LSTM cell first concatenates the h_{t-1} to the input X_t. If this prediction is the first in the sequence, the previous prediction is simply a zero vector the size of the prediction layer. This concatenated vector is then sent to a feedforward network referred to as the "forget gate" to produce f_t, which is multiplied by the cell state such that the cell "forgets" irrelevant information. The concatenated vector is then sent through two more feedforward networks, an "input gate" that outputs i_t and a "candidate cell state" network that produces C'_t. These values are multiplied to each other and added to the cell state to create a new cell state, C_t. Finally, the concatenated vector is sent through another neural network that produces O_t, and is multiplied by a pointwise *tanh* operation applied to the cell state to produce a new prediction h_t. The new prediction and cell state are passed to the next iteration of the LSTM cell with the next input data in the sequence [11].

3 Results

3.1 Feedforward Neural Netowrk

The feedforward neural network provides preliminary results for possible predictive dust event models using neural networks. Below is a table of results obtained from neural networks containing varying layers of 25 nodes with 10 principal components as inputs.

Accuracy

Hidden Layers	1	2	3	4	5	6	7	8
Training	88.5827	85.8268	84.4488	84.7441	86.4173	84.7441	84.7441	89.4685
Testing	80.0399	76.6467	73.8523	73.8543	73.8523	75.4531	75.4531	83.2335

Although it appears from the above table that the 8-layer neural network provides the most accurate results, it does not necessarly provide the most meaningful. Below, the table of true positives guessed in each neural network shows us that the 6- and 7-layer neural network correctly guesses the greatest number of dust events in training, and the 3-layer in testing.

True Positives

Hidden Layers	1	2	3	4	5	6	7	8
Training	7	47	49	55	62	67	67	0
Testing	3	37	48	39	31	46	46	0

It is also worth mentioning that as the number of true positives increases, so does the number of false positives. This finding may indicate that the neural network is discovering a pattern that occurs both in dust events and some other type of weather event, or that weather patterns found during dust events also occur many days before or after dust events as well. The table below shows the number of false positives guessed with each number of hidden layers.

False Positives

Hidden Layers	1	2	3	4	5	6	7	8
Training	16	64	71	79	73	113	113	0
Testing	19	74	95	91	78	95	95	0

3.2 Recurrent Neural Network with LSTM

With a resolution of 24 hours, the recurrent neural network will achieve 89.4685% accuracy with the existing dataset. When observing the model's guessing pattern, it is found that this level of accuracy is not meaningful as even with the weighted loss function, the RNN guesses each event as non-dust and the accuracy is equal to the proportion of non-dust events in the dataset. In order to create a model with more meaningful results, a dataset with a resolution of 24 hours would be more suitable. The training data used for this model does not have every successive event in the same location, so the LSTM cell makes a prediction using cell and hidden state information from a location possibly hundreds of kilometers away. In order to make a more useful LSTM model, a dataset with a resolution of 1 hour instead of 24 and beginning and end times of every dust event would allow for training based on 24-hour sequences for each recorded date instead of reading the entire list of events as a single sequence.

4 Conclusion

With training data that has a time resolution of 24 hours, a multilayer perceptron provides a preliminary prediction for dust events using weather forecast data. In order to use a predict dust events using an LSTM recurrent neural network, however, a greater resolution of training data may be necessary to make an hourly prediction for a specific resolution over one day. It may also be possible to create an RNN that remembers the location of the previous event, or adds the distance between the previous and current event in its cell state. With the existing data, however, more possible neural network models could be created. For instance, instead of predicting events based off of a single point on a raster, an recurrent or convolutional neural network could be implemented using the entire raster for a dust date to recognize features of a dust events.

References

[1] UNEP GEAS. "Forecasting and early warning of dust storms." UNEP. Feb. 2013. Web. May 24 2017. `https://na.unep.net/geas/archive/pdfs/GEAS_Feb2013_DustStorm.pdf`

[2] Benedict, Karl et. al. "Final Benchark Report." *Public Health Applications in Remote Sensing.* Sep. 2008. Web. May 24 2017. `http://phairs.unm.edu/publ/Final%20Benchmark%20v14b%209-30-08.pdf`

[3] Armenta, Rebecca B. "Geopotential height patterns at 500mb associated with dust storms in the United States/Mexico border region during January-May of 2011-2014." May 2016 New Mexico State University. Access May 31 2017.

[4] Tan, P. Steinbach, M. Kumar, V. *Introduction to Data Mining.* 2006 Pearson. Access May 23 2016.

[5] "GNU Wget 1.18 Manual." GNU Project. Web. Jun. 27 2017. `https://www.gnu.org/software/wget/manual/wget.html`

[6] "Rapid Refresh (RAP)." National Centers for Envrionmental Information. NOAA. Web. Jun. 27. 2017. `https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/rapid-refresh-rap`

[7] "pygrib documentation." Github. Dec. 29 2014. Web. Jun. 14 2017. `http://jswhit.github.io/pygrib/docs/`

[8] Zaremba, W. Sutskever, I. Vinyals, O. "Recurrent Neural Network Regularization." arXiv. Feb. 19 2015. Web. Jul. 5 2017. https://arxiv.org/pdf/1409.2329.pdf

[9] "Getting Started with TensorFlow." TensorFlow. Web. Jul. 5 2017. https://www.tensorflow.org/get_started/get_started

[10] LeCun, Y. et al. "Efficient BackProp." Yann LeCun's Home Page. http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

[11] Olah, C. "Understanding LSTM Networks." *Colah's blog.* Aug. 27 2015. Web. Jul. 20 2017. http://colah.github.io/posts/2015-08-Understanding-LSTMs/