Project Report

CME 250A

Pengfei Gao and Tim Moon

June 8, 2016

1 Date Processing

We investigate several machine learning models to predict temperature based on date, location, and weather conditions. These models are trained with a data set from the National Oceanic and Atmospheric Administration. This data set ranges from 1929 to 2016 and consists of 139 million daily weather reports from weather stations throughout the world. We are interested in predicting average temperature based on the following features:

Feature	Type	Units	Comments
Station ID	Categorical		Concatenation of DATSAV3 ID and WBAN ID
Year	Numeric		
Month	Numeric		
Day	Numeric		
Dew Point	Numeric	°F	
Sea Level Pressure	Numeric	mbar	
Station Pressure	Numeric	mbar	
Visibility	Numeric	mi	
Mean Wind Speed	Numeric	kn	
Maximum Wind Seed	Numeric	kn	
Gust Speed	Numeric	kn	
Precipitation	Numeric	in	
Precipitation Report	Categorical		Indicates procedure to measure precipitation
Fog	Categorical		Boolean
Rain	Categorical		Boolean
Snow	Categorical		Boolean
Hail	Categorical		Boolean
Thunder	Categorical		Boolean
Tornado	Categorical		Boolean

2 Implementation

The data set was randomly split into training (70%) and validation (30%) sets and the training set was used to train ordinary least squares, generalized linear, gradient boosting, and random forest estimators. This was implemented with H2O to allow for quick prototyping. In particular, H2O implements distributed data structures and algorithms, allowing us to easily leverage parallel computer architectures. Experiments were performed using four EC2 instances from Amazon Web Services.

3 Results

The mean squared errors for each method are reported below:

Model	Train MSE	Train R^2	Validation MSE	Validation R^2
Ordinary Least Squares	52.7	0.91	52.7	0.91
Generalized Linear Model	99.7	0.83	99.6	0.83
Gradient Boosting	138.0	0.76	137.8	0.76
Random Forest	45.3	0.92	43.8	0.92

We see that the random forest method yields the best performance. We hypothesize that the gradient boosting method struggled since the data set exhibits behavior that is too complicated to be captured by shallow trees. Although the performance is improved by incorporating multiple trees, it may take an excessive number to achieve a good model. We also note that ordinary least squares outperformed the generalized linear model, likely since it used 26716 predictors compared to the generalized linear model's 28.[1] Inspecting the random forest, we find that the most important features are as follows:

Feature	Scaled Importance
Dew Point	1
Station ID	0.16
Snow	0.12
Sea Level Pressure	0.09
Month	0.08

Dew point is a measure of humidity, which indicates that temperature is very sensitive to humidity. The standardized coefficients computed by ordinary least squares are as follows (excluding non-Boolean categorical features):

Feature	Standardized Coefficient
Year	0.576
Month	0.0760
Day	0.0025
Dew Point	20.5
Sea Level Pressure	-1.69
Station Pressure	0.0035
Visibility	1.377
Mean Wind Speed	0.244
Max Wind Speed	0.449
Gust Speed	-0.467
Precipitation	-0.436
Snow Depth	-0.424
Fog	-3.04
Rain	-3.14
Snow	-5.29
Hail	1.23
Thunder	-0.07
Tornado	-1.10

Observe that the temperature is positively correlated with the year, which is suggestive of global warming.

Appendix A Python Script

```
import csv
import h2o
```

[1]Recall that categorical features must be converted to multiple numerical features before applying a generalized linear model. Thus, it appears that the generalized linear model discarded most of the information involving station ID.

```python
import sys
orig_stdout = sys.stdout

# filepath = '/Users/Pengfei/Documents/data/'
filepath = 's3n://stanford-cme250a/weather/data/'
filelist =[ filepath+'Xheader.csv' ]
for year in range(1940,2017):
        filelist.append(filepath+'X'+str(year)+'.csv')

# Initialize H2O
# h2o.init()
h2o.init(ip="localhost",port=55555,strict_version_check=False)

# Load data from files
data = h2o.import_file(filelist)

##############data processing#################
# Convert Boolean data to categorical
data['fog'] = data['fog'].asfactor()
data['rain'] = data['rain'].asfactor()
data['snow'] = data['snow'].asfactor()
data['hail'] = data['hail'].asfactor()
data['thunder'] = data['thunder'].asfactor()
data['tornado'] = data['tornado'].asfactor()

# Delete unnecessary columns
data = data.drop('temp␣cnt')
data = data.drop('dewpoint␣cnt')
data = data.drop('sea␣cnt')
data = data.drop('stat␣cnt')
data = data.drop('visi␣cnt')
data = data.drop('wind␣speed␣cnt')
data = data.drop('*is␣hourly␣max')
data = data.drop('*is␣hourly␣min')

# Combine WBAN and DAVSAT3 station IDs to get unified station IDs
StationIds = 100000*data['Station'] + data['WBAN']
StationIds = StationIds.asfactor()
StationIds.set_name(0, 'StationId')
data = data.cbind(StationIds)
data = data.drop('Station')
data = data.drop('WBAN')

# Obtain date
months = data['MonthDay'] // 100
days = data['MonthDay'] % 100
months.set_name(0, 'Month')
days.set_name(0, 'Day')
data = data.cbind(months)
data = data.cbind(days)
data = data.drop('MonthDay')

# Remove entries with missing temperature data
data[data['temp']>9999,'temp'] = None
```

```
# data = data.na_omit()

# Remove missing data
data[data['dewpoint']>9999,'dewpoint'] = None
data[data['sea␣level␣pres']>9999,'sea␣level␣pres'] = None
data[data['station␣pres']>9999,'station␣pres'] = None
data[data['visibility']>999,'visibility'] = None
data[data['mean␣wind␣speed']>999,'mean␣wind␣speed'] = None
data[data['max␣wind␣speed']>999,'max␣wind␣speed'] = None
data[data['gust␣speed']>999,'gust␣speed'] = None
data[data['max␣temp']>9999,'max␣temp']= None
data[data['min␣temp']>9999,'min␣temp']= None
data[data['precipitation']>99,'precipitation'] = 0
data[data['snow␣depth']>999,'snow␣depth'] = 0
##############data processing#################


# Generate train set and validation set
[train, val] = data.split_frame(ratios=[0.7])

# set chosen feature
feature_list = list(data.names)
feature_list.remove('temp')
feature_list.remove('max␣temp')
feature_list.remove('min␣temp')


# Training Models

glm = h2o.estimators.glm.H2OGeneralizedLinearEstimator(model_id='glm1')
glm.train(y = "temp", x = feature_list, training_frame = train, validation_frame = val
orig_stdout = sys.stdout
outputFile = open('log.txt', 'a')
sys.stdout = outputFile
print "===============================␣Generalized␣Linear␣Model␣====================
print glm
outputFile.close()
sys.stdout = orig_stdout

gbm = h2o.estimators.gbm.H2OGradientBoostingEstimator(model_id='gbm1', distribution='g
gbm.train(y = "temp", x = feature_list, training_frame = train, validation_frame = val
orig_stdout = sys.stdout
outputFile = open('log.txt', 'a')
sys.stdout = outputFile
print "================================␣Gradient␣Boosting␣============================
print gbm
outputFile.close()
sys.stdout = orig_stdout

# Try grid search
from h2o.grid.grid_search import H2OGridSearch
hyper_parameters = {'ntrees':[50], 'max_depth':[3,5], 'learn_rate':[0.01,0.1,0.05]}
gs = H2OGridSearch(h2o.estimators.gbm.H2OGradientBoostingEstimator(distribution='gauss
gs.train(y = "temp", x = feature_list, training_frame = train, validation_frame = val)
```

```
orig_stdout = sys.stdout
outputFile = open('log.txt', 'a')
sys.stdout = outputFile
print "================================␣grid␣search␣================================"
print gs
outputFile.close()
sys.stdout = orig_stdout

rf = h2o.estimators.random_forest.H2ORandomForestEstimator(model_id='rf1')
rf.train(y = "temp", x = feature_list, training_frame = train, validation_frame = val)
orig_stdout = sys.stdout
outputFile = open('log.txt', 'a')
sys.stdout = outputFile
print "================================␣Random␣Forest␣=============================
print rf
outputFile.close()
sys.stdout = orig_stdout

#deep learning is extremely slow, might not include it in the big data version
dl = h2o.estimators.deeplearning.H2ODeepLearningEstimator(model_id='dl1')
dl.train(y = "temp", x = feature_list, training_frame = train, validation_frame = val)
orig_stdout = sys.stdout
outputFile = open('log.txt', 'a')
sys.stdout = outputFile
print "================================␣Deep␣Learning␣=============================
print dl
outputFile.close()
sys.stdout = orig_stdout

# data=h2o.get_frame('Xheader.hex');
# glm2 = h2o.h2o.get_model('glm-37c7dbc7-c3a0-4c48-8dd7-693e9a4c9cfe')
# orig_stdout = sys.stdout
# outputFile = open('log.txt', 'a')
# sys.stdout = outputFile
# print "================================ Generalized Linear Model ==================
# print h2o.model.model_base.ModelBase.coef(glm2)
# outputFile.close()
# sys.stdout = orig_stdout
```