# Large Scale Learning

STAT 672

APRIL 21, 2021

SCOTT HOYLAND

# What is Large Scale Learning?
## Data Size

- Sometimes, Large Scale Learning is defined by the memory size of the data
  - Any ML done on data above 1GB is large scale learning
- Another definition is Learning done on data that doesn't fit in memory
  - Most computers have a RAM of 8GB or 12GB
  - The popular GPU, *GEFORCE RTX 3090*, has 24GB of memory
- The size of data being collected continues to increase with datasets in the TB or PB range becoming more common.

# What is Large Scale Learning?
## Computational Cost

- Along with datasets being too large to fit in memory, another consideration in Large Scale Learning is the computational cost of performing analysis on these large data sets.

- Learning algorithms may be too expensive in terms of computing time with large data sets

- Imagine an algorithm that requires $O(D^2N)$ flops (linear regression)
  - On a data set of size 1,000,000x1000 (approximately 1GB file size) has computational time of 1e12 FLOPS (1 TFLOPS)
  - On a data set of size 1e9x1000 (approximately 1TB file size has computational time of 1e15 FLOPS (1 PFLOPS)

- Best GPUs have computational power of approximately 36 TFLOPS (0.03s, 27.8s)
- Higher end CPUs have computational power of approximately 250 GFLOPS (4s, 4000s or 66.7 minutes)

# How to Perform ML on Large Data Sets

- Options for performing analysis on large data sets:

1. Obtain more memory and/or processors with a higher FLOPS
2. Bootstrap Sampling
3. Stochastic Gradient Descent
4. Distributed (Parallel) Learning

# Pros and Cons of Methods 1-3

- Obtain more memory or better processors
  - For datasets in the smaller scale of Large Scale Learning, this is a simple solution that can utilize any learning technique used on smaller data sets.
  - Processors and memory have a finite amount of scaling.

- Subsampling of Data
  - This option can reduce the number of used observations to manageable levels.
  - This option can increase bias and may lead to overfitting if the feature set is large.

- Stochastic Gradient Descent
  - For large-scale problems, second order stochastic gradient and averaged stochastic gradient are asymptotically efficient after a single pass on the training set (Bottou).
  - Stochastic Gradient does not address the memory problem.

Bottou, Leon. *Large-Scale Machine Learning with Stochastic Gradient Descent.* NEC Labs America.

# Pros and Cons of Distributed (Parallel) Learning

- Pros
  - Addresses the computational problem of Large Scale Learning by allowing different components across a local network or across the internet to coordinate to perform ML.
  - Allows scaling “horizontally” meaning adding more computers or servers to the network.
  - Data can be stored in smaller “chunks” across the network.
  - Increased reliability due to members of the distributed network being able to pick up tasks from a failed piece of the system.

- Cons
  - Writing and running a distributed machine learning algorithm is highly complex.
  - Not all algorithms are easily parallelized.
  - A distributed algorithm must spend resources “communicating” across members of the system.

# Some Common Distributed Learning Options

- Writing your own distributed learning code is often not necessary outside of novel problems
- Packages in Python
  - PySpark
  - RevoscalePy
  - Dask
- Packages in R
  - Rpubs
  - Sparklyr
  - RevoscaleR
- Other options:
  - CUDA
  - Apache Hadoop

# Spark in R with sparklyr

- sparklyr does not read the data into the R environment memory.
- Instead, it creates a mapping in R to the data location.
- The data is only read into R when it is needed
- dplyr (standalone or within tidyverse) is the best package to use for data manipulation with sparklyr
- The machine learning functions (ml_*) in sparklyr are distributed functions

- Set up instructions using RStudio:

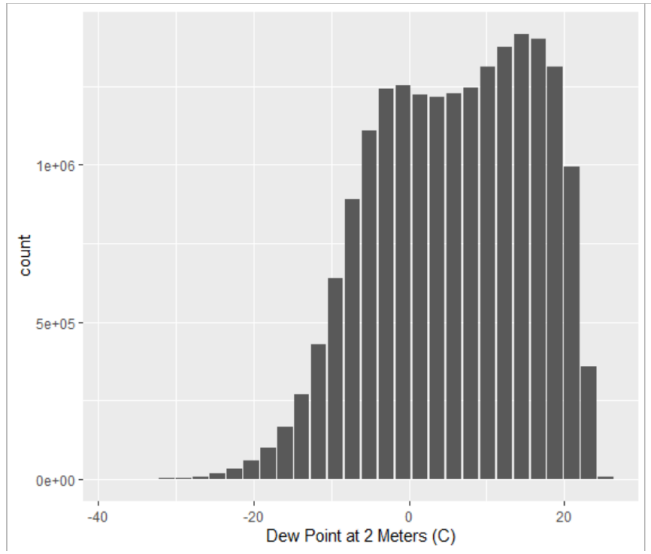
```
install.packages("sparklyr")
install.packages("tidyverse")
install.packages("installr")
library(sparklyr)
library(tidyverse)
library(installr)
spark_install(version = "3.1.1")
install.java(version = 11,
        page_with_download_url = "http://jdk.java.net/java-se-ri/",
        path = "C:/java")
sc <- spark_connect(master = "local")
```

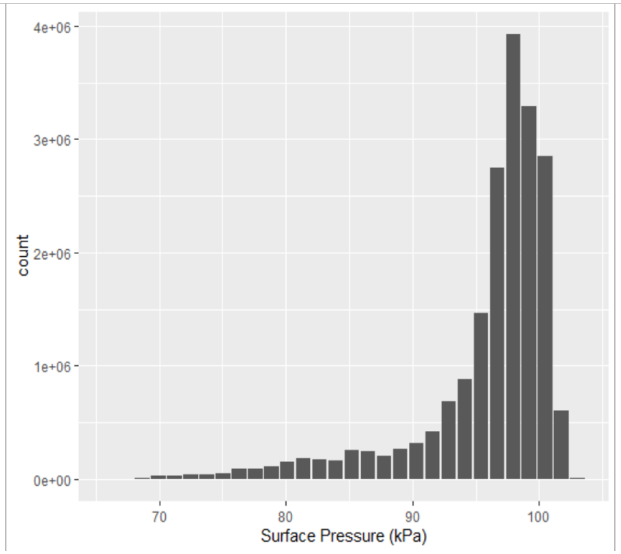
# Machine Learning Example - Data

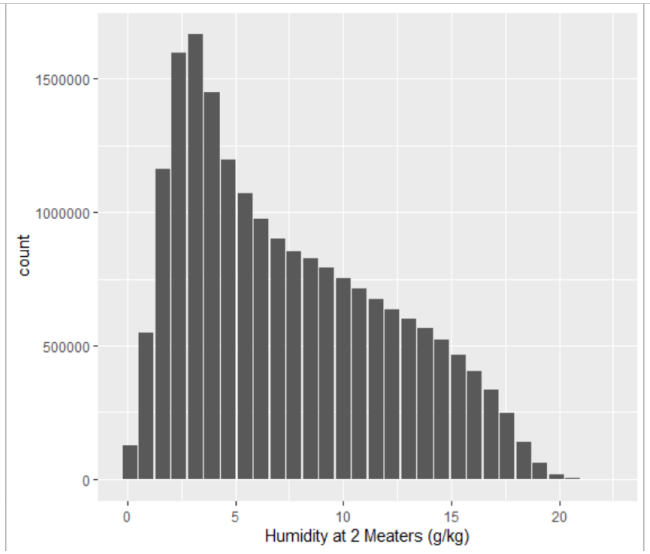
- Everyday, people turn on the television or look online to see what the day's weather will be like. Meteorologists spend countless hours attempting to create accurate forecasts. One aspect of the weather forecast is predicting the amount of rain (if any) on a given day.
- Using a dataset containing a wide range of meteorological data, I am trying to predict the total precipitation on any given day using linear regression. Also, I am trying to predict if any day will have rain using logistic regression.
- The original dataset contains 21 variables with 19.5 million observations in the training set and 2.3 million observations in each of the the test and validation sets.
- The size of the training data is about 2GB, and the size of each of the test and validation data is about 250MB.
- A data mapping was created using sparklyr::spark_read_csv for the training and the test data.

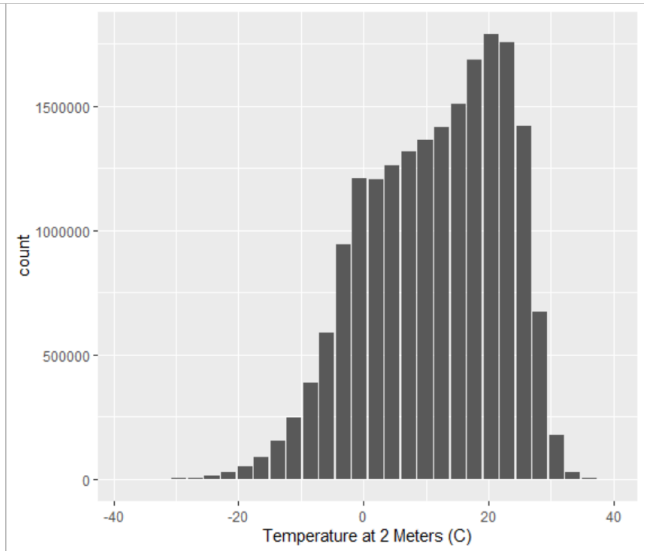
# Machine Learning Example - Plots

- The R package “dbplot” is a bridge between sparklyr and ggplot. The package addresses the data using Spark mapping and manipulates it into a ggplot figure.
- Th following are histograms of four of the data points representing dew point, surface pressure, humidity, and temperature



# Machine Learning Example - Results

- Performed a linear regression on the data to try to predict the amount of rainfall in a day.
  - R-Squared of 0.1
  - Linear regression took 4.5 minutes to run
- Performed a logistic regression on a binary variable where 1 = day had precipitation and 0 = day had no precipitation.
  - AUC > 0.8
  - Logistic regression took 8 minutes to run
- When I tried to run the same models using base R for comparison, it took over 12 minutes to read in the data, and there was not enough memory available to run linear regression using the lm() function

# Sparklyr - Takeaways

- PROS
  - Allows for data to reside in hard drive/SSD until it is needed for manipulation, visualization, or calculation with only a mapping in RAM
  - Creates an interface between R and Spark, where Spark functions can be run in R and R functions can be run on Spark data frames
  - Machine Learning Algorithms are written for distributed/parallel calculations.

- CONS
  - Requires learning a syntax that is neither fully consistent with R nor fully consistent with Spark.
  - Not a widely used package so documentation and online resources are scarce compared to other R packages
  - Errors are often based on Java code so can be difficult to identify bugs
  - Can cause R to stall or crash