CS230

County-Level COVID-19 Case Predictions using Deep Learning

William Buchanan
wbuchan@stanford.edu

Kento Perera
kperera1@stanford.edu

Timothy Sah
tsah@stanford.edu

Shannon Yan
shannony@stanford.edu

Abstract

This project aims to train a deep neural network to accurately predict the number of COVID-19 cases in a US county based on demographic and historical case data as well as state-wide policy data. We trained a feed forward 5-layer neural network as well as a LSTM network on data from 2,257 US counties. Our goal is to get accurate time-series predictions of cases as well as analyze which features have the largest impact.

1 Introduction

The problem we are investigating is predicting the number of COVID-19 cases in a given county based on relevant factors from recent COVID data. We aggregated data from the C3.AI Covid-19 Data Lake which contains many different data points from organizations such as John Hopkins University, World Health Organization, The New York Times, etc. The input features we are using for each county range from demographic data to Covid cases/deaths data to state-wide policy data and more. We then use a Keras feed forward neural net with an Adam Optimizer with the number of neurons corresponding to the number of input features as well as a separate LSTM model. The $\hat{y}$ value we initially predicted was the number of positive cases on 2020-10-10. However, we extended our model to predict the number of new cases for 3 weeks into the future in 5 day intervals. This project is relevant in understanding the projected magnitude of a new COVID outbreak given the county's information. As a result, this will allow counties to be better informed and prepared for a potential outbreak as well as see how implementing different COVID policies can affect their projection.

2 Related Work

COVID-19 prediction algorithms have been a major focus in recent moths, leading to hundreds of papers being published in the field. We first read *Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM* to get a comparison of the most effective predictive models and found that LSTMs and Bi-LSTMS provided the highest accuracy. Within the category of deep learning using LSTMs there have been many recent papers published, most notably *DeepCOVIDNet: An Interpretable Deep Learning Model for Predictive Surveillance of COVID-19 Using Heterogeneous Features and Their Interactions* - IEEE Journals Magazine. This paper uses spatial time series data to predict cases on the county level for a future 7-day period. Similarly, *Prediction and analysis of COVID-19 positive cases using deep learning models: A descriptive case study of India*, uses LSTMs to generate time series predictions. However, this model only takes raw case data into account and

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

falls short of the approach of the previous paper, which also incorporated static features and multiple time series. We built off the work DeepCOVIDNet by curating a dataset with more case data and substantially more static features to try to make accurate time series predictions for a larger time horizon of 25 days versus 7 days.

3 Dataset and Features

We have 2,357 training examples and 115 test examples. Each example consists of a single county in the United States and has 121 features. A table containing a summary of features can be found under Figure 1 in the appendix.

We obtained our data via the C3.ai COVID-19 API (https://c3.ai/covid-19-api-documentation/) which allows us to pull the latest data surrounding COVID from sources such as the John Hopkins University, the COVID Tracking Project, and The New York Times as well as demographic information on the county level from the US Census Bureau, the Bureau of Labor Statistics, and other sources. We combined two types of data: static and time series.

The static data includes snapshots of county data for all features except for case and death count. For current policy information, the C3.ai API sourced information from the Kaiser Family Foundation's Social Distancing Policies data. We used one-hot encodings to represent policy implementations by county. We also obtained data on the United States general election 2016 presidential results by county from Townhall.com. To prepare the data to be fed into our network, we concatenated demographic, case, and political datasets and filtered out counties with missing values.

We used new COVID deaths and case counts over 5 day periods in each county from June 1st 2020 to November 8th 2020. We chose a 5-day interval as it helps filter out some of the day-to-day noise in the data and allows us to efficiently forecast out relatively far without having to make predictions for a large number of time steps. While we had two time series (cases and deaths), we decided to only make predictions on case data.

4 Methods and Architectures

4.1 MultiLinear Regression (Baseline) and Feed Forward Neural Net

For our initial learning model, we took inspiration from the project, *House Price Prediction Using Deep Learning*, since both our projects attempt to learn the weights of a multitude of input features so it can output a single numerical prediction.

Our first step is to split the data into training and testing sets. We chose a 90/10 training to testing ratio after seeing our Neural Net was under-fitting on a 66/33 ratio.

Before we input our data into either our Baseline or Deep Learning model, we first scale our input features using the Standard Scaler function from scikit-learn.preprocessing. The feature scaling works by standardizing features by subtracting the mean and dividing by the variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set.

For our baseline model, we used MultiLinear Regression which attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y. Formally, the model for multiple linear regression, given n observations for p explanatory variables, is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ...\beta_p x_{ip} + \varepsilon_i \text{ for i = 1,2, ... n}$$

For the loss function we are trying to minimize, we use the least-squares model where the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0).

To analyze the accuracy of our predictions, we calculate the Root Mean Square Error (RMSE) which represents the square root of the sum of the differences between predicted values and observed values.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Our second main metric that we use to measure accuracy is the variance score from scikit learn where the best possible variance score is 1.0 and lower values are worse. The scikit learn variance score is calculated by

$$variancescore(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)}$$

Next, for our Deep Learning method, we use a Keras Sequential Neural Net Model from Tensorflow. For our final model features, we use 119 neurons where each neuron corresponds to an input feature and our model consists of 5 densely connected layers where a Dropout layer with p=0.2 follows the first 2 dense layers. Our final layer was a densely connected layer with 5 output neurons corresponding to the 5 most recent data points we were attempting to predict. In addition, we use ReLU activation at each layer and use an Adam optimizer. For our loss function, we use Mean Square Error which has been shown above, excluding the square root. While training the model, we split the training data into mini-batches with size of 128 and ran the model for 400 epochs. We then used the same evaluation metrics as the Linear Baseline Model of RMSE and variance score to compare our results.

4.2 Long Short-Term Memory (LSTM) Model

To expand upon our initial learning architecture, we began exploring LSTM Model architectures to capture the time-series components of the COVID-19 case count data on a per-county basis. We first defined our problem as a multi-step time-series forecasting problem[2], and then reconfigured our data to focus solely on dynamic, time-sensitive data (case counts within five-day windows) rather than static data. After this, we converted our time-series data to fit a supervised learning problem, with inputs consisting of case counts at $t - n, ..., t - 1, t$ time steps, and target values consisting of case counts at the $t + 1, ..., t + 5$ time steps.

We constructed our LSTM model through the Keras Sequential class, and it consisted of a stateful LSTM layer with one neuron, followed by a Dense layer with an output shape of five, representing case count predictions at each of the following $t + 1, t + 2, ..., t + 5$ time steps. As with our Feed Forward Neural Net Model, we used the Mean Square Error loss function and Adam optimizer.

To train our LSTM Model, we used a batch size of one, reserved 10 prediction sequences for testing purposes, and then trained over 50 epochs. As with both our MultiLinear Regression Model and Feed Forward Neural Net Model, we evaluated our results with RMSE and variance.

5 Experiments/Results/Discussion

5.1 MultiLinear Regression (Baseline) and Feed Forward Neural Net

For our initial results with our Feed Forward Neural Net model, we achieved a Root Mean Square Error of 1300.89 with a variance score of 98.92%. This slightly outperformed our baseline of a multiple linear regression model which had a Root Mean Square Error of 468.32 and a variance score of 99.86%.

However, we found that these results were unexpectedly high as well as that the baseline linear model outperformed our Deep Learning model. We realized that a better problem to solve was predicting the number of new cases per day, rather than the cumulative number of cases since the data points would be less linear/more complex. By reformatting our data and changing the predicted output, we found that our Deep Learning Network scored considerably higher than our linear baseline model in both the RMSE and variance score metrics explained in the prior section.

The hyperparameter choices we made in our Feed Forward NN structure were the number of hidden layers, the number of neurons, using Dropout, and the activation function. We initially started with a 3-layer network but through experimentation, we found that increasing the depth to 5 layers resulted

in better performance. The number of neurons was determined by the number of available features in our data set, which was 119. Using manual experimentation with Dropout, we found that adding a Dropout layer with p=0.2 after the first and second dense layers optimized our performance. And finally, relu activation was the better activation function than sigmoid or softmax.

The hyperparameters we chose relating to our training algorithm were the learning rate optimizer, the number of training epochs, and the mini-batch size. We used an Adam optimizer to optimize our learning rate and decreased our number of training epochs to 200 from 400 since the cost function converged fairly early. For our mini-batch size, we found that 128 worked the best between 32, 64, and 256.

For our final results with our Feed Forward model, we were able to achieve a RMSE of 136.82 and a variance score of 83.11%. This significantly outperformed our baseline multi-linear regression model which had a RMSE of 181.69 and variance score of 71.33%.

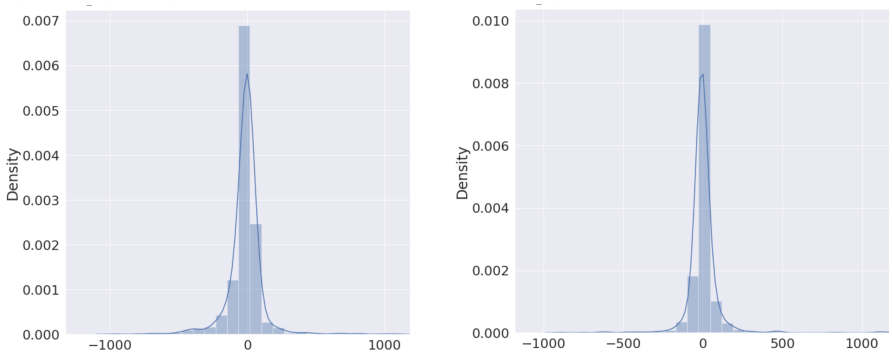

Figure 1: Left: Residual Plot of Multi Linear Regression vs Right: Residual Plot of Keras Neural Net

The two plots above show a density plot of the residuals calculated by the actual y value minus the predicted y value. We see that the Keras Neural Net has a higher density when the residual is near 0, showing that the NN performs better.

5.2 Long Short-Term Memory (LSTM) Model

With our county-specific LSTM Model, we focused on evaluating results from a model fit to make predictions for future cases in Santa Clara County. In terms of hyperparameter tuning, the limited number of data points motivated us to choose a batch size of 1. Furthermore, our experiments revealed that optimal results were relatively consistent as we varied the number of neurons in our LSTM Layer, so we ultimately decided to stick with a single-neuron layer to achieve faster training. Finally, we initially attempted to train over a larger number of epochs, but found that training over 50 epochs was a sufficient balance to ensure we weren't over-fitting or under-fitting.

The RMSE and variance evaluation metrics for our LSTM Model produced evaluation scores over predictions at each of the $t+1, ..., t+5$ time steps. We observed the following RMSE and variance scores:

Time Step	RMSE	Variance
$t+1$	29.20	0.73%
$t+2$	43.51	0.69%
$t+3$	73.65	0.54%
$t+4$	76.80	0.52%
$t+5$	100.24	0.44%

As expected, prediction accuracy decreased as we tried to predict case counts further into the future. Though our evaluation metric does not perfectly align with that of our Feed Forward model and multi-linear regression model, we see that the LSTM Model performs respectably against both. A diagram graphing the output of our LSTM predictions can be found under Figure 2 in the appendix.

5.3 Feature Correlation

One aim of our project was to explore the correlation between our input features and the number of cases in a given county, with specific emphasis on state-wide policies. Unsurprisingly, we found that population size and proxies for population size such as ICU beds, unemployed population, etc. had the highest correlation to a county's Covid cases. For this reason, we excluded all population size related features and found the correlation between each feature and the number of new cases on Nov 8th, 2020 (the most recent data point). Interestingly, we found that the percent democrat had the highest correlation to new cases whereas percent gop had the greatest negative correlation to new cases. Qualitatively, we believe this can be caused by the fact that a greater number of gop-dominated counties are rural and have smaller county populations, resulting in fewer covid cases. Another surprising find was that policies such as a large gatherings ban and closing restaurant dine-in had positive correlations to new cases whereas policies such as no mask requirement or no stay-at-home policies had negative correlations to new cases. However, we believe this is caused by the fact that this policy data was recorded very recently on Sept 12th and that counties struggling with covid cases would impose stricter policies than counties with relatively few cases.

6 Conclusion/Future Work

Based on our evaluation metrics of RMSE and variance for the number of new cases per day, the LSTM model greatly outperformed our baseline model and presented a 26% improvement in RMSE over the feed forward neural network. This was expected, as regression and basic neural networks have limited capacity to understand temporal data compared to LSTMs.

COVID-19 case prediction is an important task that has massive implications for policy planning and governance, but defies to be easily addressed with existing models and datasets. Because of the time it takes to implement new policies and distribute medical resources, the 15-25 day predictions are the most critical; however, our model's performance steadily dropped with each time step, ending with a variance of 44% with an RMSE of 100.24 for predicting cases 25 days in the future. A cursory examination of the body of COVID-19 prediction algorithms reveals that there are no gold-standard approaches to the problem. Among the 21 modeling groups selected by the CDC to model cases on the national and county levels, the variance in predicted trajectories is very wide. Despite this massive range of predictions, the CDC website notes that "Over the last several weeks, more reported cases than expected have fallen outside of the forecasted prediction intervals. This suggests that current forecast prediction intervals may not reflect the full range of future reported case numbers. Forecasts for new cases should be interpreted accordingly."

With this context, our results are unsurprising; however, there is much that can be improved on to boost the accuracy of our time series predictions. First, an increase in our computing resources would allow us to leverage our data more effectively using methods such as multivariate LSTM models to incorporate COVID-19 fatalities times series data into our case prediction. More compute power would also enable us to use data that would capture the relationship between counties. For instance, we had access to a dataset of movement between US cities which we could use to link data between counties. This fell outside the scope of this project due to the sheer volume of data and size of the model that this would necessitate, but would be a good starting place for continuation of this project.

7 Contributions

Will Buchanan: Researched related works, handled c3.ai API calls, and cleaned/compiled dataset

Kento Perera: Worked on finding relevant Neural Nets, LSTM models, and baseline models and adjusting those models to fit our data. Researched analysis metrics as well as tuned hyperparameters and analyzed performance.

Tim Sah: Merged initial dataset features, reconfigured dataset to fit cases-per-day framework, and tuned and experimented with LSTM models.

Shannon Yan: Worked on accessing policy data via C3.ai API and prepossessing this data such that it would be an adequate input for model.

References

[1] A. Ramchandani, C. Fan and A. Mostafavi, "DeepCOVIDNet: An Interpretable Deep Learning Model for Predictive Surveillance of COVID-19 Using Heterogeneous Features and Their Interactions," in IEEE Access, vol. 8, pp. 159915-159930, 2020, doi: 10.1109/ACCESS.2020.3019989.

[2] J. Brownlee, "Multistep Time Series Forecasting with LSTMs in Python," 27-Aug-2020. [Online]. Available: https://machinelearningmastery.com/multi-step-time-series-forecasting-long-short-term-memory-networks-python/. [Accessed: 17-Nov-2020].

[3] Parul Arora, Himanshu Kumar, Bijaya Ketan Panigrahi, "Prediction and analysis of COVID-19 positive cases using deep learning models: A descriptive case study of India", in Chaos, Solitons Fractals, Volume 139, 2020, doi: 10.1016/j.chaos.2020.110017.

[4] Zhang, Xuan, et al. "AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction." IOP Conference Series: Materials Science and Engineering. Vol. 569. No. 5. IOP Publishing, 2019.

[5] Farah Shahid, Aneela Zameer, Muhammad Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM, Chaos", Solitons Fractals, Volume 140, 2020, 110212, ISSN 0960-0779, https://doi.org/10.1016/j.chaos.2020.110212.

[6] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[7] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[8] Chollet, F., others. Keras. GitHub. Retrieved from https://github.com/fchollet/keras, (2015).

[9] Mahsa Mir, "House Prices Prediction Using Deep Learning", towardsdatascience.com, July 21, 2020.

[10] Jason Brownlee, "Multistep Time Series Forecasting with LSTMs in Python", machinelearningmastery.com, May 10, 2017.

[11] Jason Brownlee, "Multivariate Time Series Forecasting with LSTMs in Keras", machinelearningmastery.com, August 14, 2017.

Appendix

Feature	Description
hospitalIcuBeds	The total number of hospital ICU beds within the county.
hospitalStaffedBeds	Total number of staffed hospital beds.
hospitalLicensedBeds	Total number of licensed hospital beds.
latestTotalPopulation	Most recent population of the county based on data from The World Bank or the US Census Bureau.
latestLaborForce	Most up-to-date labor force population of the location based on available Bureau of Labor Statistics data.
latestEmployedPopulation	Most up-to-date employed population of the location based on available Bureau of Labor Statistics data.
latestUnemployedPopulation	Most up-to-date unemployed population of the location based on available Bureau of Labor Statistics data.
latestUnemploymentRate	Most up-to-date unemployment rate of the location based on available Bureau of Labor Statistics data, in percent.
populationCDS	Population of the location, based on data from Corona Data Scraper.
per dem	% of county who voted Democratic in the 2016 election based on data from Townhall.com.
per gop	% of county who voted Republican in the 2016 election based on data from Townhall.com.
stayAtHome-Lifted	0 or 1 to represent policy in place
stayAtHome-NoAction	0 or 1 to represent policy in place
stayAtHome-Statewide	0 or 1 to represent policy in place
stayAtHome-Rollback	0 or 1 to represent policy in place
....	....
faceCoveringRequirement-GeneralPublic	0 or 1 to represent policy in place
faceCoveringRequirement-AllowOfficialsGeneralPublic	0 or 1 to represent policy in place
faceCoveringRequirement-None	0 or 1 to represent policy in place
newCases5DaysPrev2020-06-06	New covid cases over 5 days prior to 2020-06-06.
newDeaths5daysPrev2020-06-06	New deaths over 5 days prior to 2020-06-06.
newCases5DaysPrev2020-06-11	New covid cases over 5 days prior to 2020-06-11.
newDeaths5daysPrev2020-06-11	New deaths over 5 days prior to 2020-06-11.
.....	.....
newCases5DaysPrev2020-11-08	New covid cases over 5 days prior to 2020-11-08.
newDeaths5daysPrev2020-11-08	New deaths over 5 days prior to 2020-11-08.

Figure 1: Features and Descriptions

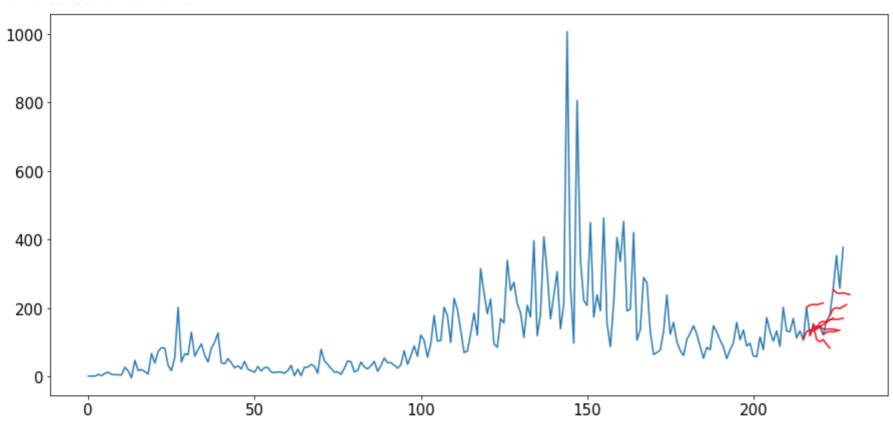

Figure 2: LSTM Model Predictions (in red)