Credit Suisse Research Project

Pranav Rupireddy

Lab Notebook

\*Along the way I try to bold potential improvements to be tracked and pivots/main/summary points

1/3: First Step: Research Features/Literature Review

**Planned Approach: 1) Research Features 2) Develop Simple Regression Model 3) Develop More Complex Model**

Explanation: I don’t have too much awareness around the features that are to be used in unemployment prediction, so the first step will be researching this. Then, given that I’m imagining the final model to be relatively complex, it will be a good idea to get started with a baseline (the first model), though, since it will be a good source of comparison and will make the final task of step 2 less intimidating as step 1 could somewhat serve as a starting base and in case step 2 fails. This second model may have to get broken into iterative changes.

Brookings Study (https://www.brookings.edu/wp-content/uploads/2012/09/2012b\_barnichon.pdf)

-Two Main Approaches:

A. Historical Times-Series Properties of the Unemployment Rate & Near-Term Indicators of the Labor Market

B. Okun’s Law (Relationship between Output Growth and Unemployment Changes)

C. New Approach: Incorporate Information on labor force flows

-3 months for a shock to lead to convergence between conditional steady state (unemployment rate if current rates continued) and true (law of motion for this convergence based on diffeq)

-forecasts the flows (autoregressed w the two flows + w two other features: vacancy posting and initial claims for unemployment insurance)

-t+j flows, t+j rate is weighted average of prev and conditional state state rate (law of motion again more or less with the predicted instead of actual for the prev period) using the rate as weight

-3 state case: more flows & diff laws of motion (b/c diff diff eqs based on flows necessarily), but same idea of using this law in conjunction with auto-regressed flows

-2 state the quantities are inferred indirectly and 3-state the quantities are published (A to B rate is A to B in time t divided by A in t-1)

-Comparison: 2 Professional (Greenbook – benchmark forecast, Median from SPF), 3 Alternates (ARIMA autoregressor, VAR with labor force flows, initial claims and help-wanted, and one where you hold the rates constant)

-especially outperforms standard model during onset of recession (because can capture unemployment asymmetry like a harsh unemployment with flow model)

-especially great performance in the near term

-combined performance is the best (OLS of the models)

-3-state better than 2-state in the near term

-Okun’s Law: 2% drop in GDP associated with a 1% increase in unemployment (<https://www.investopedia.com/terms/o/okunslaw.asp>)

-Checked with another paper (<https://www.atlantafed.org/-/media/documents/research/publications/wp/2015/01.pdf>) and it appears as if the main models corroborate (professional, autoregressive models, and flow-based – in fact they use real gdp growth (Okun’s) for their flow model here) and seems to corroborate the improvement from combination and from a near-term advantage of flow-based models

1/5: Second Step

**The work in 1/3 has basically concluded the first step. Here is an updated plan:**

**Updated Plan: 1) Literature Review (DONE), 2) Develop Simple Autoregressed Model, 3) Develop more complex model potentially integrating GDP growth and flows**

Explanation: **The initial plan was to develop a simple regression model for the first plan but it looks like the standard baseline is the simple autoregressive model.** The more complex model will combine multiple approaches that were mentioned in the two papers above. I’ll think more deeply about the second model later but I’ve chosen flows and GDP growth since the flow predictions features won’t be as useful since we are just predicting the near-term US unemployment rate. GDP growth is an add-on mentioned in the Brookings paper that I don’t believe was used in that paper’s primary study

Data –

* Keeping the next step in mind, it looks like the Brookings study uses flows from BLS after 1990, 1967-1976 from Joe Ritter (asks reader to see Shimer 2012), 1976-1990 from Nekarda’s database (though this I think is from longitudinally matched Current Population Survey)
* The second paper uses the Current Population Survey for flows and then uses Federal Reserve Bank of Philly Real-Time Data Research Center for measure of real GDP
* Seasonal Adjustment? It looks like from the Brookings paper that seasonal adjustment did not affect much, so that shouldn’t matter
* CPS is from the Bureau of Labor Statistics
* Unemployment data itself appears to be from BLS

**For this step, at least, I’ll work off of the 2022-12-02 vintage of UNRATE (data from 1948), the monthly unemployment rate published by the CPS/BLS.(**[**https://alfred.stlouisfed.org/series?seid=UNRATE**](https://alfred.stlouisfed.org/series?seid=UNRATE)**), slightly modified for readability into a notebook (See C). I’ll use CPS data for flows.**

Model – I’ll use **AR(6)** since the Federal Reserve Bank of Atlanta paper seems to state that this can serve as a baseline standard. Since I’m planning on firing off (at least) two models, it’d be nice to have multiple models all side-by-side. Therefore, I’ll use **Jupyter Notebooks**.

Originally, I was under the impression that the model could, when testing, could be trained and fix its parameters off of a single training session and then make predictions on a new time series data using these parameters. It looks like, at least for Python, that it will just forecast off of the given train data. So, instead, I’ll probably fire off many independent train/tests. **I’ll use RMSE since that’s what the two papers above use.**

Results – After noticing high levels of variability and inaccuracy (I created naïve prediction method that just used the previous time period – this outperformed the trained predictor; the literature also reported higher performance levels) with random out of sample tests, I decided to just predict all but the first 100 time stamps. **I noticed that my RMSE (.03263) is much lower than the RMSE that is reported in the papers (~.20). This is concerning**. But at least, it is better than the naïve (.049333). This should be something that may be apparent as I proceed, or otherwise, if there is no time left over, I could set is a potential future direction.

1/6-10: Third Step (Part 1)

**Updated Plan: 1) Literature Review (DONE), 2) Develop Simple Autoregressed Model (DONE), 3) Develop more complex model**

1. **Integrating GDP growth**
2. **Flows**

**I have instead decided to break down the final model step into two sub steps to make this more manageable.** In the first substep, I add in GDP growth. In the second substep, I add in the labor flows.

Data – I’m planning to use the Brave-Butters-Kelley Gross Domestic Product (https://fred.stlouisfed.org/series/BBKMGDP). **I did notice that it is not the most updated** (it has three months behind). However, I can’t seem to find a more updated GDP growth dataset. Most datasets appear to be quarterly. In terms of production, I think this serves as more so a Proof-of-concept model. When moving to production, it appears as if there are monthly datasets elsewhere online. Also, it looks like Brave-Butter-Kelley uses monthly data (https://www.chicagofed.org/publications/chicago-fed-letter/2019/422); I’m sure these data could be use to create a proprietary measure of monthly GDP growth. At the worst-case, this could just get dropped out of the model; there are other features in the model. **Unfortunately, while that dataset is published by the St. Louis fed, it looks like it isn’t in the published datasets that we are supposed to pull from.** **So, instead, I’ll be using this dataset:** <https://fred.stlouisfed.org/series/A191RL1Q225SBEA>. This, too, is lagged, so the same caveats apply to this dataset. For this dataset, I’ll need to join on data, and then I’ll probably just drag the data for the non-matched months. I’ll have to first check that all the data are matched up. I’ll use training time series of 6 to match the AR(6).

Modeling – I’ll probably plan on changing the architecture but something along lines of the code laid out in this article is probably the best to go: https://medium.com/mlearning-ai/multivariate-time-series-forecasting-using-rnn-lstm-8d840f3f9aa7. This is the use of multivariate input to predict a univariate time series, which matches up with my use case. **LSTM.**

Another thought that came up once I was testing this forecasting model was the idea of out-of-sample testing on the dataset I am using for this. I realized that for the autoregressive models, I just use the last data point, whereas for the LSTM models, I do a full 80/20 % train-test split. Theoretically, I could do the same strategy where I test just the final data point, but it would take just too much time to 800 or however main hyperparameter training sessions. It is true that for the regular, non-hyperparameter optimized model, I may have the time to do these many trains, but it would be nice to keep some level of uniformity between the hyperparameter and non-hyperparameter optimized models. A related idea is that there is fundamentally different time ranges used. I am rationalizing this away once again because the model architectures are different: neural networks are meant to be done in a 80/20 fashion but the Brookings paper seems to test autoregressive framework rather liberally with testing.

Results – I created a preliminary run of the more complex model. The RMSE is very high (~.7). After adding in a validation + early stopping it lowered a bit but not by much (~.6). **The LSTM model’s error is higher than that of the AR model.**

*Hypotheses*

1. Logic error in my code,
2. The way I am splitting up my train and validation (with regards to scaling: do I reuse a scaler trained on just training set or train for the whole thing?)
3. The prediction and if I am using scaling properly here (use new or old scaler). I’m getting really good losses on training data. Alternative to scaling/abandoning it altogether?
4. It’s possible that the gdp growth data isn’t the best because of the missing values potentially (so consider dropping this and see performance; **a more long-term approach to this would be to investigate different missing value imputation methods**)
5. It’s possible I need the hyperparameter optimization framework, so I could move on to this step
6. It’s possible I need to line up the test set with the AR test set.
7. I noticed that that the training and (in particular) validation losses were great, so I thought that maybe the issue was that the validation dataset was unrepresentative of the full population.
8. It’s also possible that I just need a different model altogether.

*Experiments*

* #1: Couldn’t find a logic error
* #2 & #3: Took out the scaling and it worsened (1.2 – the updated wouldn’t be too much better)
* #4: Took out the GDP growth parameter and it worsened (Updated: .966)
* **Aside: I also realized that the mean squared error metrics that I was using was actually squared. I took the square root and the RMSE for AR makes more sense now (~.139), though it’s still a bit low.** The naïve is .21.
* Aside: Reran (there may have been some parameter changes) but original run is .934. I should have been tracking which parameters I’m using.
* #6: The AR on the last part of the data is even lower
* #5 & 7: tried doing a hyperparameter optimization and the rmse completely fell: 4.34. For this, I decided to do use k-fold cross validation (https://www.geeksforgeeks.org/hyperparameter-tuning-using-gridsearchcv-and-kerasclassifier/). This is because K-fold cv would solve this issue given that it uses parts from all of the data. I looked through the hyperparameter optimization code. I didn’t see any major issues. The only thing I noticed was that there was a tensorflow retracing warning. It looks like, though, that that is more of runtime issue than it is a performance issue ([Better performance with tf.function  |  TensorFlow Core](https://www.tensorflow.org/guide/function#controlling_retracing)). **I think another potential research avenue is to look at other hyperparameter optimization frameworks**, especially considering that the rmse should not have dropped. Here is an example: <https://www.tensorflow.org/tutorials/keras/keras_tuner>.
* #8: Given that after removing the gdp growth column the model still didn’t do well, this means that the LSTM is probably less good of a fit than the autoregressive models. What remains is more or less the same data as for the more basic model (AR - 6); just the vanilla time series of unemployment. Therefore, I went ahead and tested out VAR (chose this model because it was the one used in the brookings study and because it is a simple extension of the AR model). I used anywhere from 1-12 lags since that was what was used in the brookings study. This article was helpful (https://cprosenjit.medium.com/multivariate-time-series-forecasting-using-vector-autoregression-3e5c9b85e42a). I ended up getting a value of .245, which is better than the LSTM, but is worse than AR (I double checked AR on this dataset too and ended up getting the same RMSE). **This confirms the hypothesis: autoregressive models seem to just work better on this type of data.**

The concludes the first part of the third step.

1/12 – 1/15: Third Step (Part 2)

Data – As I discussed in part A of the second part, I am going to plan on using the labor flows data from the CPS. **Just a note here, I’m noticing a spike in a lot of these datasets around the pandemic time. I’m wondering if some sort of smoothing could be applied here (Kalman Filter etc.).**

I realized that all of the labor force flows are not in the published datasets, so I just went ahead and published my own data list. Since I was at it already, I decided to publish that monthly gdp growth dataset.

**Hypothesis: I’ll check first that the monthly dataset has higher performance.**

**Given that it probably (and in fact does) will (see bolded imputation idea above) and given that I wanted to use this at first, I think it’s worth retesting the prior frameworks with this dataset instead and using this one moving forward**. **Anyways for a proper apples-apples comparison this is the best thing to do**.

Very conveniently, when downloading this data list, it outer joined all of them. I just did some work in excel to calculate the probabilities. **I’m using CPS data for flows, as mentioned above.**

Results –

Experiment for Hypothesis:

New GDP & Unemployment with VAR: .065

Old GDP & Unemployment with VAR: .074

**This confirms the hypothesis.**

All with VAR: .285 (Used VAR here because as stated above it does better than LSTM and this will be shown below with the new dataset as well)

*Retest of prior frameworks*

GDP & Unemployment with VAR: .065

Technically this dataset is slightly different than the dataset used to test just the unemployment with AR model, so I’ll actually retest that one too.

AR(6): .023

Naïve: .071

LSTM (Unoptimized) GDP + unemployment: 1.84

LSTM (optimized) GDP+ unemployment: 2.25

This is all of the unit changes (though more tests are possible) to test certain features:

Naïve -> AR(6) -> VAR (with GDP) -> LSTM with GDP -> Hyperparameter optimized (with GDP) LSTM

-> VAR with all

-> VAR with old GDP

There theoretically there are more tests possible, but I’ll stop here. AR(6) performs the best overall. **This runs against the literature** which seems to believe that the multivariate auto models will do better. It also still is very low compared to the reported accuracy. But, in a sense, it confirms the reliability of the model as a baseline. I wonder if adding more variables actually makes the model worse.

**Summary of Potential Future Improvements:**

1. **Filters/smoothing**
2. **Other hyperparameter optimization frameworks**
3. **Performance tension with Literature**