refactor_math_PhD

March 15, 2020

1 Case Study: Math PhD Exchange

This data set is derived from the article "Eigenvector-Based Centrality Measures for Temporal Networks" by Taylor, Myers, Clauset, Porter, and Mucha, which was in turn scraped from the Mathematics Geneology Project. There is an entry $i \to j$ in year t if a mathematician graduated in year t from university j and thereafter supervised at least one PhD thesis at university i. This is a proxy for hiring data: we are assuming, essentially, that this mathematician was hired ("endorsed") by i in or around year t.

1.1 Data Limitations

There are several limitations in these data implied by the collection process.

1. If a mathematician is hired in year t, then they are unlikely to show up in the data set until roughly year $t + \tau$, where $\tau \approx 5 \pm 2$ is the approximate amount of time required to complete a PhD thesis.
2. If a mathematician is hired sequentially by departments i and i', both i and i' will be taken to have endorsed this mathematician *in the same year* t, even though the actual time in which i' hired the mathematician may be well after i. This raises the potential for causality violations.

To address limitation 1., we omit the final six years of the data set. We'll study the years 1960-2000.

2 Data Preparation

First we read the data. In order to ensure connectedness of the endorsement matrix across many time periods, we restrict the data to only the top 100 schools by placement.

[2]:
```
((61, 70, 70),
 array([1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956,
        1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967,
        1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978,
        1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989,
        1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
        2001, 2002, 2003, 2004, 2005, 2006]))
```

In order to ensure that the endorsement matrix is weakly connected at the initial condition, we are going to take all of the pre-1960 data and aggregate it into the initial state matrix. In order to address limitation 1) above, we are also going to exclude the years 2001-onward from analysis.

[3]:
```
array([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970,
       1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981,
       1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992,
       1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000])
```

3 Inference

Now we're ready to try to learn the model on some data. In the paper, the instantaneous rate of endorsements to j is given by $\gamma_j \propto e^{\beta s_j}$. This rate is independent of the identity of the endorser.

The model used here is an extension in two respects. First, we use a more general rate γ that depends on the identity of both the endorser and the endorsed:

$$\gamma_{ij} \propto e^{\beta_1 s_j + \beta_2 (s_i - s_j)^2}$$

The incorporation of the quadratic term in the exponent governs a tendency for agents to endorse each other when they are nearby in the hierarchy. If β_1 is negative, endorsements that span large swaths of rank-space are discouraged. For example, in the context of faculty hiring, negative values of $-\beta_2$ would discourage cases in which a very low-ranked school was able to hire a candidate from a very high-ranked school. The model that we have been considering can be recovered by fixing $\beta_2 = 0$.

Second, we also consider a version of this model in which, instead of s_j denoting the SpringRank of j, we set $s_j = \sqrt{d_j}$. In this case, it doesn't matter "where" you are in the hierarchy, only the total number of endorsements you've received.

The result of SpringRank-based inference is:

```
computing memory hyperparameter lambda
computing parameter vector beta
```

[4]:
```
{'lam': array([0.8989368]),
 'beta': array([ 2.97285196, -1.10612226]),
 'beta_stderr': array([0.03453119, 0.03713385]),
 'LL': -21040.66924207652}
```

The result of degree-based inference is:

```
computing memory hyperparameter lambda
computing parameter vector beta
```

[5]:
```
{'lam': array([0.89095018]),
 'beta': array([ 1.46669029, -0.2453722 ]),
 'beta_stderr': array([0.01570473, 0.01639991]),
 'LL': -20578.89819718238}
```

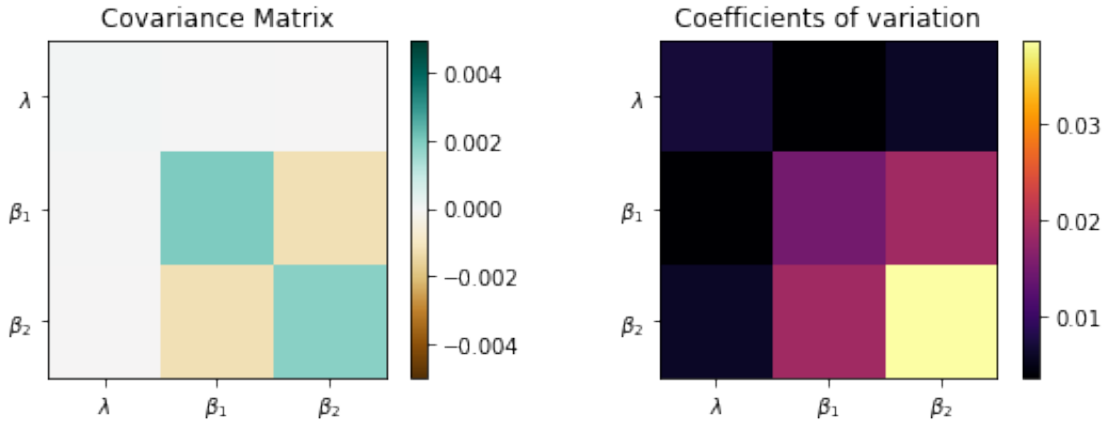
These results suggest that the degree-based feature model may be preferable to the SpringRank model, based on its higher log-likelihood (LL). Other points to note are that the models give very similar estimates for λ, indicating that they agree on the relevant timescales for prediction in these data.

Although the degree-based result appears better, we're going to proceed with the SpringRank-based model for now.

4 Parameter Uncertainty

Now let's estimate the covariance matrix of the parameters by inverting the Hessian matrix of the likelihood function at the parameters we solved for.

```
/Users/philchodrow/Dropbox
(MIT)/projects/!side_projects/prestige_reinforcement/py/features.py:13:
RuntimeWarning: overflow encountered in exp
  gamma = np.exp(phi)
```



The model is extremely confident about its prediction in λ, indicated by the fact that the rows and columns corresponding to λ are very small. There is a negative correlation between β_1 and β_2, indicating that there are regions of parameter space that are "nearly as good" (as measured by the likelihood) in which β_1 is smaller and β_2 is larger, or vice-versa.

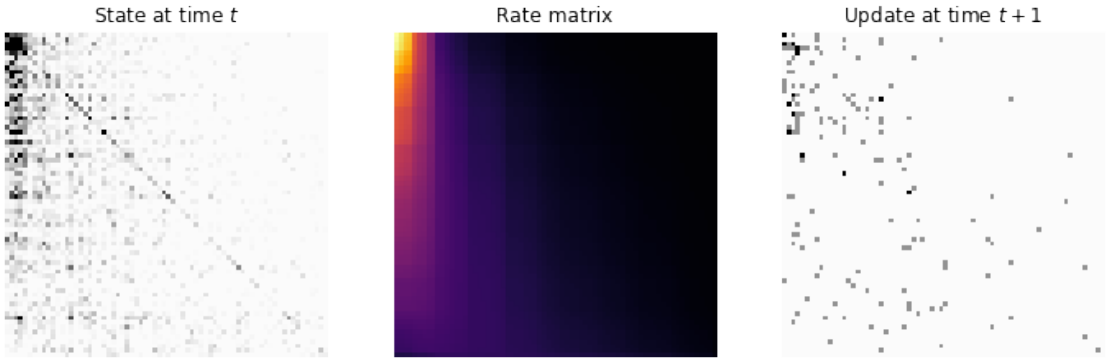
How "big" is the covariance matrix relative to the data? Not too large! The coefficients of variation for each pair of variables are all well below 5%, indicating that we can be roughly 95% confidence of our estimates within 10% accuracy or better. The greatest amount of relative uncertainty is in the estimate of β_2.

While this is a highly informal approach and should not be confused with actual statistics, the small coefficients of variation are heuristic evidence that all the parameters are significantly different from zero.

5 Snapshot of model prediction

The mini-study below considers how the model ''works."

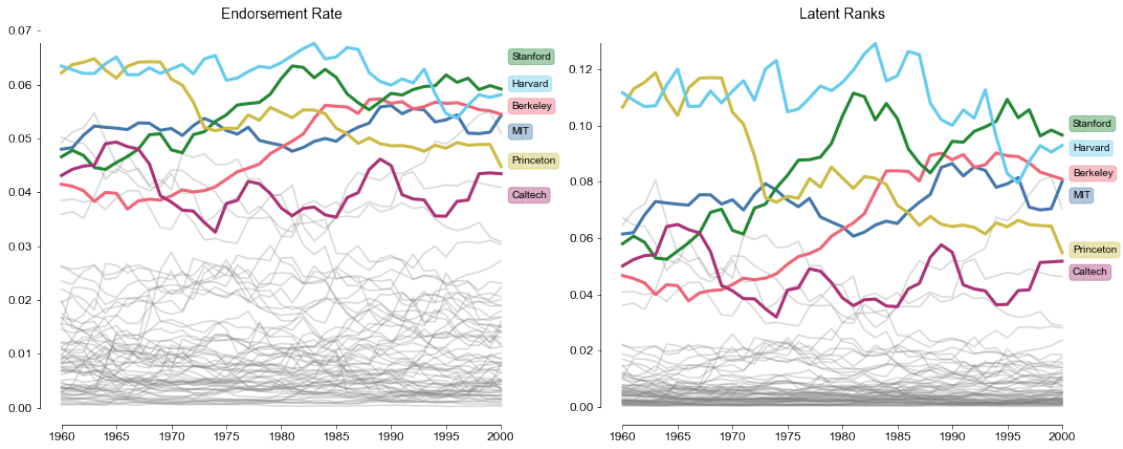
[29]: `(-0.5, 69.5, 69.5, -0.5)`



On the far left, we show the model state A_t at time $t = 30$. Each entry of this matrix is a weighted sum of previous hiring events: $A_{ij}^{(t)} = \sum_{\tau=1}^{t} \lambda^{\tau} E_{ij}^{(\tau)}$, where $E_{ij}^{(\tau)}$ is the number of times i endorsed j in time period τ. This matrix is an input into the rate matrix Γ, which is shown in the center. This is a matrix of predictions about endorsements in the next time-step: entry γ_{ij} is proportional to the predicted probability that i will endorse j in the next timestep. Finally, we can (heuristically) compare the rate matrix to the observed update at time $t+1$, noting that the bulk of endorsement events do indeed occur in regions of high probability specified by the rate matrix.

6 Inferred rank dynamics

Now let's visualize the evolution of this system as understood by the model. It's of possible interest to visualize at least two distinct objects. The first is the overall modeled endorsement rate, which predicts the rate of endorsements of j in the next timestep. The second is the intrinsic ranks modeled for each individual school.



These two figures tell an interesting story. On the left, the endorsement rate is proportional to the modeled rate at which a given school will be endorsed (i.e. place a candidate) in the next timestep. There is considerable heterogeneity in the ranks, as we would expect.

If we look over to the right, however, we see much stronger inequality. That's because the righthand side has controlled for peer-effects, and is estimating pure prestige scores without regard for what schools are nearby in the rankings. These scores are more directly interpretable as pure measures of the prestige of each school.

Mathematically, the endorsement rate on the righthand side is given by the expression $\gamma_j = \sum_i \gamma_{ij}$ where

$$\gamma_{ij} \propto e^{\beta_1 s_j + \beta_2 (s_i - s_j)^2}$$

as before. The inferred ranks are

$$r_j \propto e^{\beta_1 s_j} \ .$$