Cross-sectional State-Space Forecasting with Partial Pooling

Mike Bedard
Economic Technology
Amazon
Seattle, WA 98019
mbedard@amazon.com

Matt Johnson
Topline Forecasting
Amazon
Seattle, WA 98019
jnmmatt@amazon.com

Paul Sangrey
Topline Forecasting
Amazon
New York, NY 10018
psangrey@amazon.com

Abstract

We propose a novel architecture for time series models built upon state-space methods. We jointly estimate many, potentially multivariate, distributions defined using state-space models by partially pooling their parameters across the cross-section. These joint distributions define a novel recurrent neural network. By combining state-space methods and neural networks, we leverage the interpretability of state-space models and the scalability and flexibility of neural networks. This lets us build an accurate, flexible, and scalable forecast that is not a black box. We implement this architecture by building a library on the deep learning library MXNet to leverage state-of-the-art scalable optimization techniques including automatic differentiation and computation graphs to estimate the parameters governing the state-space models. This library abstracts over a large class of state-space models allowing users to estimate almost arbitrary model specifications without code changes. We forecast weekly business formation by state, which is obtained from the FRED economic database at the St. Louis Federal Reserve bank. We show that this forecast is more accurate than state-of-the-art neural network approaches (DeepState) and more accurate than state-of-the-art univariate generalized linear models (Prophet) when the data are particularly volatile.

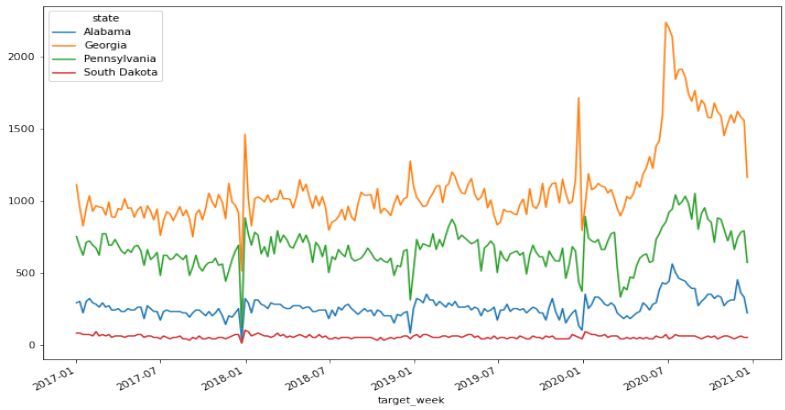
1 Introduction

Decision-makers often require forecasts for a group of noisy series that exhibit similar qualitative behavior but do not have identical dynamics. For example, our empirical application is a group of 51 state-level weekly series. The series exhibit similar seasonal patterns and overall trends (e.g., people do not usually start businesses the week of Christmas, regardless of geography). However, different states differ vastly in size and rate of growth of business formation. We plot a small segment of the data in figure 1 to provide a picture of the data's qualitative behavior.

The customers for many forecasts are humans. They are making consequential, irreversible decisions and need to know why they should trust the forecast. Being able to explain why a forecast is reacting in a certain way is of first-order importance. Recently, there has been a burgeoning literature on explainable artificial intelligence. Two leading motivations in the literature for building explainable models are trustworthiness and transferability [2]. These two benefits are closely linked because to trust a model we must know where it works well, i.e., what spaces to which we can "transfer" the model and still get accurate, informative results.

This paper proposes a hierarchical state-space model that separates the model's parameterization into two pieces. The first piece is the time-invariant (static) part. We build a novel method to partially-pool information across the cross-section in order to enable the model to learn complex, similar but not identical patterns even when some series are short and noisy. The second piece defines dynamics, which are learned in a set of potentially multivariate groups called nodes. We model the data in each

Figure 1: Business Formation Example Data



of these nodes using a conditionally linear state-space model. This creates a parsimonious representation — we do not need to learn a nonlinear transition kernel — that can be easily interpreted by our human customers. This also creates arguably the most serious limitation of our work, which is inherited from state-space models: we only support conditionally linear propagation of shocks.

The initial leading application for state-space methods in economics was forecasting macroeconomic data, which often have cross-sectional dimensions of less than ten and time dimensions in the low hundreds. The key objects of interest in that literature are the dynamics and propagation of shocks [5, 11]. Even models with substantial non-linearity (e.g., stochastic volatility) often use conditionally linear propagation mechanisms. Conversely, the neural network literature has focused on models with vast cross-sectional dimensions, but their maximum forecast horizon is typically fairly small relative to the data's time and cross-sectional dimensions. Researchers commonly jointly train neural networks on tens or hundreds of thousands of time series. For example, see various sequence-to-sequence models such as convolutional neural networks [15, 17].

Data with time and cross-sectional dimensions that are both in the dozens to low thousands has taken on renewed interest recently in both econometrics and machine learning, [3, 9, 12, 13, 16]. In the empirical application, we have 51 weekly series with 827 periods. We have used the methods here to produce anything from univariate forecasts to forecasts of thousands of series.

As in the Deep State model [12], we bridge the recurrent neural network and econometric literatures by building a network using state-space models' latent structure. This jointly satisfies the explainability, scalability, and accuracy requirements by building a state-space model that fully describes the behavior of each series given a set of parameters. We parameterize this state-space model in terms of a set of structural parameters, i.e., parameters with business-defined meaning.[1] We pool these parameters across the cross-section to learn them jointly. We do not attempt to ensure that our pooling procedure can be explained to stakeholders, just that the parameters and resultant forecasts are explainable. This flexible pooling is viewed as a neural network and learned through stochastic gradient descent, which we implement using MXNet.

We benchmark ourselves against DeepState, which is arguably our closest competitor, and Facebook's Prophet [14] because it also focuses on easily configurable and explainable generalized linear models. The key difference between our approach and DeepState is that we give the user far more control over the state-space model being estimated. In particular we let the user encode a series of structural restrictions on the state-space system. The key difference between our approach and Prophet is that we learn the time-invariant parts of the forecast using cross-sectional information leading to meaningful accuracy gains. Similar to Prophet, we develop a library that can implement a wide variety of different models with the same underlying architecture. We focus on precise esti-

[1] Note, although our model has linear propagation of shocks, it is not linear in the parameters.

mation of the human-interpretable parameters such as autoregressive coefficients, initial states, and holiday lifts.

The remainder of this document is organized as follows. Section 2 reviews state-space forecasting and discusses the general model we are estimating. It also describes how we regularize the parameters across nodes. Section 3 describes the logical design of the library, how to estimate effectively arbitrary state-space models, and the model we are estimating in the empirical application. Section 4 describes the model's empirical performance. Section 5 concludes.

2 The data generating process and state-space modeling

Let $Y := \{y_{n,t}\}_{t=1,n=1}^{N,T}$ denote a set of N time series each of length T and let $X := \{x_{t,k_n}\}_{t=1,k_n=1,n=1}^{T,K_n,T}$ denote N sets of K_N exogenously-given series. For each (potentially multivariate) endogenous time series y_n, we have a set of exogenous variables X_n and their forecasts. For example, X_n can include holiday dummies, Fourier coefficients, and time trends. We condition on these exogenous variables throughout.

Let $Z := \{z_{n,t}\}_{t=1,n=1}^{N,T}$ denote a set of N unobserved multivariate time series each of length T. Let $\Theta := \{\theta_n\}_{n=1}^{N}$ denote the parameters of series n and $\Gamma(\theta_n, X_n)$ denote a transformation of those parameters. This $\Gamma(\theta_n, X_n)$ can be arbitrarily non-linear.

Then the distribution of $y_{n,t}$ given $z_{n,t-1}$ is

$$p\left(y_{n,t}|z_{n,t-1}, \Theta, X_n\right) = \int \phi\left(y_t|\Gamma(\theta_n, X_n), z_{n,t}\right) \phi(z_{n,t}|\Gamma(\theta_n, X_n), z_{n,t-1}) dz_{n_t}, \tag{1}$$

where $\phi|\theta_n$ is a Gaussian likelihood, with parameters controlled by $\Gamma(\theta_n, X_n)$. The $\Gamma(\theta_n, X_n)$ parameters directly enter the likelihood. That is, they are reduced-form parameters. The θ_n parameters, in contrast, are structural parameters. They are parameters that stakeholders understand.

The goal is to jointly estimate the implied distributions across all N of the nodes y_n. We can explicitly model the dependence between two series by placing both series in the same y_n, which is a multivariate time series, in general. We share information across nodes in the estimation but do not estimate a full joint distribution. If N is quite large (we use $N \approx 50$ in the empirical application, though N can be hundreds or thousands in general), estimating a full joint distribution is effectively impossible.

Instead, we jointly estimate a set of marginal distributions for each of the $\{y_{n,t}\}_{t=1}^{T}$. Consequently, our log-likelihood is the sum across N of the log-likelihoods of the $\{y_{n,t}\}_{t=1}^{T}$ series. This implies the joint log-posterior $\log \mathcal{L}\left(\Theta, Z|Y, X, \right)$ is proportional to

$$\sum_{n=1}^{N}\sum_{t=1}^{T}\Big(\log \phi\left(y_{n,t}|\Gamma(\theta_n, X_n), z_{n,t}\right) + \log \phi\left(z_{n,t}|\Gamma(\theta_n, X_n), z_{n,t-1}\right)\Big) + \sum_{n=1}^{N} \log g\left(\Theta, Z_n|X_n\right), \tag{2}$$

where $g(\Theta, Z|X) = \sum_{n=1}^{N} \log(\Theta, Z_n|X_n)$ is the joint conditional prior for the (θ_n, Z_n), and is assumed to be separable across Z_n. As the X_n may be overlapping, we lose no generality in letting it only depend on X_n. Because the Z_n are unobserved from a Bayesian perspective they are additional parameters, over which we specify a prior.

In particular, we construct $g(\Theta, Z|X)$ to partially pool information across nodes n. The goal is to precisely estimate Θ by assuming that θ_{n_1} is similar but not identical to θ_{n_2}, for $n_1 \neq n_2$.

2.1 The likelihood

This subsection focuses on a single node (set of series) and describes the likelihood. Adapting the notations of [6], state-space models are defined using two equations. First, is the *observation equation*, which relates the latent states to the observed series

$$y_{n,t} = \mu_t(\theta_n, X) + D_t(\theta_n, X) z_{n,t} + \epsilon_{n,t} \tag{3}$$

where D_t is the *design matrix* containing coefficients (loadings) which may be known or estimated, μ_t is the observation intercept, and ϵ_t is a vector of forecast innovations for the observed variables, where $\epsilon_{n,t} \overset{indep.}{\sim} N(0, \Sigma_t(\theta_n, X))$. The latent vector $z_{n,t}$ is also called the *state vector*.

The second equation is the *transition equation*. This equation governs the dynamics of the latent states:

$$z_{n,t+1} = \nu_t(\theta_n, X) + T_t(\theta_n, X) z_{n,t} + R_t(\theta_n, X) u_{n,t} \tag{4}$$

where consecutive state vectors are related by the *transition matrix* T_t, which contains loadings that may be known or estimated, $u_{n,t}$ is a vector of innovations in the predicted state, and the *selection matrix* R_t is included so that we may assume the conditional covariance matrix of $u_{n,t}$ has full rank without loss of generality. We assume that $u_{n,t} \sim N(0, \Omega_{n,t})$. We collectively refer the matrices that appear in the equations above as *state-space matrices*. The θ_n are the structural parameters and Γ is a functional mapping $\theta_n \to \{\mu_{n,t}, D_{n,t}, \Sigma_{n,t}, \nu_{n,t}, T_{n,t}, R_{n,t}, \Omega_{n,t}\}_{t=1}^{T}$. This is the link function in the language of indirect inference, [8]. Although the state-space matrices can depend upon θ_n, X_n, and t, the state-space conditions on the state-space matrices. The link function Γ must be deterministic, but can be highly nonlinear in general. Also, we ignore any variability in $x_{n,t}$ below. All of our inference is conditional on θ_n and $x_{n,t}$. Also, although (2) defines a posterior distribution, we use maximum a posteriori (MAP) estimation and do not draw from this posterior.

2.2 The prior

As mentioned above, the goal is to precisely estimate Θ by assuming that θ_{n_1} is similar but not identical to θ_{n_2}. We do this by penalizing deviations between the θ_n. This function behaves analogously to LASSO or ridge penalties. These penalties can be viewed as both a prior in a MAP estimation or just as a penalty term with desirable properties.

Certain structural parameters are difficult to estimate using data from a single node; thus, we want to share information across nodes. This is completely analogous to Bayesian hierarchical modeling. We draw each of the θ_n from some hyper-distribution, and this hyper-distribution adds an additional term to the posterior. This setup is extremely general because it only requires exchangeability assumptions across N, which are often satisfied if we model the $x_{n,t}$ correctly, [10].

In the empirical example, we do not model dependence between Z. For simplicity, we exclude that in the discussion below. In general, we can specify $g(\theta, X)$ using any commonly-used neural network architecture. We could use multi-layer perceptrons, convolutional neural networks, dense networks, and so on. That literature finds this architecture works well in practice, as dense layers often do, [4]. In the interest of interpretability we continue the Gaussian likelihood based approach. In particular, we adapt the implied covariance matrix from the matrix normal distribution. We do not use any more sophisticated neural network approaches such as attention or forget gates. More sophisticated and complex architectures are an interesting avenue for future research.

We let $g(\theta_n, X_n)$ take the form:

$$g_1(\theta, X) = \frac{N}{2}\log\det(\Lambda) + \frac{P}{2}\log\det(\Omega) - \frac{\lambda_1}{2}\operatorname{Tr}(\mathbb{C}\text{ov}_\Omega(\Gamma(\theta_n, X_n))\Lambda) - \frac{\lambda_2 P}{2}||\Lambda||_1 - \frac{\lambda_3 N}{2}||\Omega||_1, \tag{5}$$

where the λ values allow flexibility in weighting different pieces of the penalty function, and both Λ and Ω are positive-definite hyper-parameter matrices. We need to pick the $\lambda_1, \lambda_2, \lambda_3$ parameters. They control the estimation of Λ and Ω, which are themselves hyperparameters governing the regularization of Θ.

Equation (5) contains five terms. The first two constrain the overall size of Λ and Ω; the last two enforce sparsity through penalties on their L_1 norms. The trace term penalizes the spread of the θ parameters. Borrowing intuition from the matrix normal distribution, we define $\mathbb{C}\text{ov}_\Omega(\Gamma(\theta_n, X_n))$ as the empirical covariance of $\Gamma(\theta_n, X_n)$ based on N "observations" weighted according to the $N \times N$ matrix Ω. That is we have a weighted covariance of the reduced-form parameters. The penalty (5) has two main motivations. The first is the graphical lasso for Λ, the parameter precision matrix [7]. The graphical lasso is one of the leading algorithms for sparse inverse-covariance (precision) matrix estimation. Also, (5) bears a striking resemblance to the matrix-normal log-density for a demeaned matrix $\Gamma(\Theta, X)$ with left-covariance matrix Λ, and right-covariance matrix Ω:

$$-\frac{NP}{2}\log(2\pi) + \frac{N}{2}\log\det(\Lambda) + \frac{P}{2}\log\det(\Omega) - \frac{1}{2}\operatorname{Tr}(\mathbb{C}\text{ov}_\Omega(\Gamma(\theta_n, X_n))\Lambda), \tag{6}$$

Equations (5) and (6) share three common terms. The first two are the log determinants of Λ and Ω. Up to scale, these are identical. The third term is the negative trace of $\mathbb{C}\text{ov}_\Omega(\Gamma(\theta_n, X_n))\Lambda$, where Λ

is the prior scale. By choosing to set our matrix mean using the parameter means across nodes, we penalize parameter deviations from that mean. Our formulation allows the amount of dependence to vary across parameters and to vary across nodes.

Our prior is closely related to dense networks and Gaussian-based mean estimation. If the data are better approximated by a discrete process, this structure will struggle to pick it up. Also, we are maximizing the posterior, not drawing from it. This implies that any distributional results will not include parameter uncertainty.

3 Implementation

State-space models differ along a wide variety of dimensions. They may be univariate or multivariate, possibly with differing numbers of endogenous series. They may have various numbers of latent states to describe the dynamics of the series. The transition equation may include various numbers of independent innovations.[2]

We are interested in the parameters Θ_n. We want to estimate them to aid interpretability and to impose bounds on them when appropriate. These bounds can even include cross-equation restrictions such as stationarity. Furthermore, the functional Γ, which maps the structural parameters to reduced form parameters, can be almost arbitrarily nonlinear. Obtaining accurate forecasts, identifiability of θ_n, and trustworthiness of the stochastic gradient descent optimization limits the types of Γ we can use, but the library itself does not.

Not only do we have a specific model that implements this framework, we also provide a library that estimates essentially arbitrary state-space models. To the best of our knowledge, no existing software lets users to configure a wide class of state-space models without code changes. Statsmodels lets you inherit from a state-space object and uses numerical differentiation to maximize the likelihood. Using inheritance is a code change, and numerical differentiation does not let us scale well in either the T or N dimensions, [1]. Conversely, DeepState, [12], does not let you embed structural restrictions on the state-space matrix, making intepretability difficult.

3.1 The Model

The model being estimated in the empirical application is a fairly standard level-trend state-space model. Recall (3):

$$y_{n,t} = \mu_t(\theta_n, X) + D_t(\theta_n, X)z_{n,t} + \epsilon_{n,t}$$

The Design matrix $D_t(\theta_t, X)$ has three parts. The first controls the persistent behavior. The second loads on a cyclic (autoregressive) state. The third is a matrix of Fourier Terms. The observation intercept $\mu_t(\theta_n, X)$ equals $x_t\beta$ where x_t is a matrix of dummy variables encoding the location of holidays and β is a loading matrix that controls the percent lift from the holiday. The observation innovation covariance $\sigma^2_{\epsilon_{n,t}}$ is a scalar that is estimated and bounded below by zero.

Consider now the transition equation (4):

$$z_{n,t+1} = \nu_t(\theta_n, X) + T_t(\theta_n, X)z_{n,t} + R_t(\theta_n, X)u_{n,t}.$$

The state intercept ν_t is zero. The transition matrix $T_t(\theta_n, X)$ has three pieces. The first piece encodes level-trend dynamics $\begin{pmatrix} 1 & 1 \\ 0 & \rho \end{pmatrix}$, where ρ the persistence of the trend is estimated. The second piece encodes an $AR(4)$ process in state-space form. The third piece encodes the loadings on the Fourier terms, which are assumed to have unit roots.

The selection matrix $R_t(\theta_n, X)$ also has three pieces. The first allows for individual shocks to the trend and level. The second part encodes the single shock used by the AR(4) (cyclic) component. The third part sets all of the annual seasonal coefficients to load upon two shocks. In particular, the shocks to the cosine terms and the sine terms are each perfectly correlated. Hence, the covariance of state innovations $\Sigma_{u_{n,t}}$ has shocks for the level, trend, cyclic, sine coefficients, and cosine coefficients. All of these variances are independent and their variances are estimated.

[2]To see why each state may not require an independent innovation, consider representing an autoregressive model of order two in state-space form. The representation includes two latent states z_t and z_{t-1} where the second is a deterministic lag of the first.

Turning now to the prior, recall (5):

$$g_1(\theta, X) = \frac{N}{2}\log\det(\Lambda) + \frac{P}{2}\log\det(\Omega) - \frac{\lambda_1}{2}\mathrm{Tr}(\mathbb{C}\mathrm{ov}_\Omega(\theta_n)\Lambda)) - \frac{\lambda_2 P}{2}||\Lambda||_1 - \frac{\lambda_3 N}{2}||\Omega||_1.$$

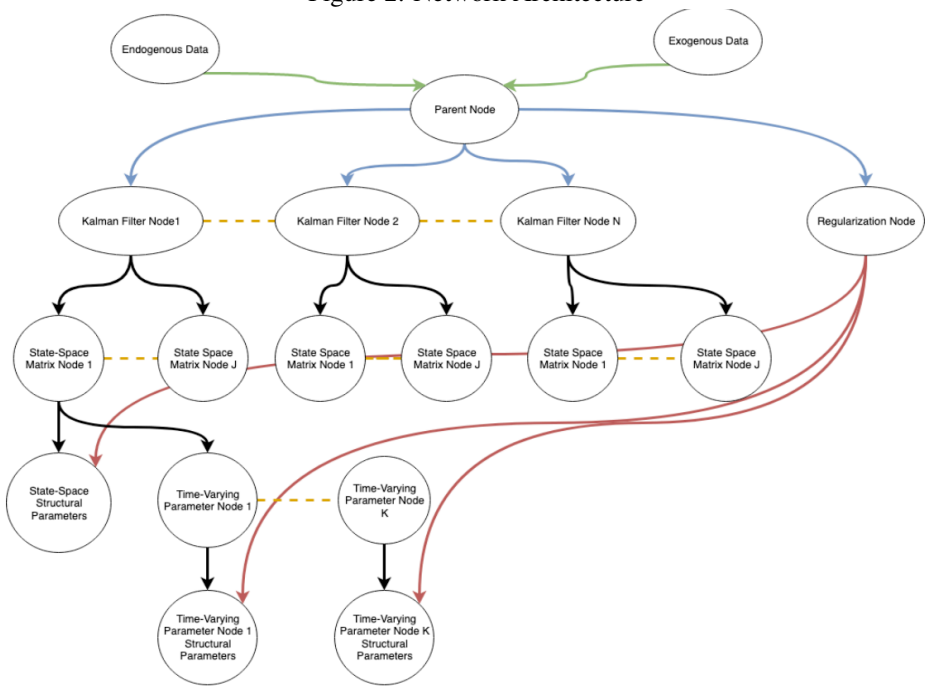
The covariance matrices (Ω and Λ) are themselves estimated by maximizing the joint posterior. We set $\lambda_1 = 0.01$, $\lambda_2 = 0.10$, and $\lambda_3 = 50.0$.

We estimate the model using Nesterov-accelerated adaptive moment estimation (Nadam). The performance of the estimator is not particularly sensitive to how it is configured. In particular, learning rates and so on are not particularly important.[3] We ran our model for 200 epochs in expanding windows every two months from January 2016 through November 2020.

We ran the model on AWS Batch backed by C5.24xlarge instances, one forecast date at a time. Each forecast took several hours (≈ 10) on a single machine. We did not attempt running the forecast on multiple machines nor running it on a GPU. Doing so would likely substantially speed up training.

3.2 Design of the Library

Figure 2: Network Architecture



To recall section 2, we have endogenous data $y_{n,t}$, exogenous data $x_{n,t}$, latent states $z_{n,t}$, structural parameters θ_n, and state-space matrices $\mu_{n,t}, D_{n,t}, \Sigma_{n,t}, \nu_{n,t}, T_{n,t}, R_{n,t}$, and $\Omega_{n,t}$. To estimate them, we need to compute the likelihood of $y_{n,t}$ and the regularization term $g(\Theta, X)$. Given the state-space matrices, we can use the Kalman filter to compute the likelihood. We automatically differentiate this likelihood with respect to these state-space matrices and differentiate them with respect to the structural parameters. We compute the state-space matrices and regularization terms as functions of $x_{n,t}$, time, θ_n, Ω, and Λ. Figure 2 graphs the network.

The one complexity in figure 2 that is not addressed above is the time-varying parameter nodes. This is how we let the state-space matrices vary with time. As described above, the regularization works on the structural parameters, (or transformations of them) not the state-space matrices.

Because the library treats state-space models as a particular case of neural networks and is implemented using MXNet, we can fully leverage the optimization and parallelization technologies available in a modern deep learning library.

[3]The parameters are base learning rate: 0.1, final learning rate: 1.0e-5, warmup steps 20, max update 200, warmup being learning rate 3.0e-4, and use a linear warmup mode.

We allow the user to specify arbitrary fixed parameters in the state-space matrices. Specifying time-varying parameters requires the practitioner to create a block that controls the parameters' time-variation. Once the block is created, the appropriate state-space matrix blocks can be modified to depend on it. We currently support time-varying 1) Fourier seasonality, 2) arbitrary linear shifts in the mean driven by exogenous variables, 3) deterministic time-of-year seasonality in the day-of-week seasonality, 4) deterministic Fourier-based time-of-year seasonality in the observation innovation covariance, and 5) arbitrary linear exogenous shifts in the observation innovation covariance.

4 Empirical performance

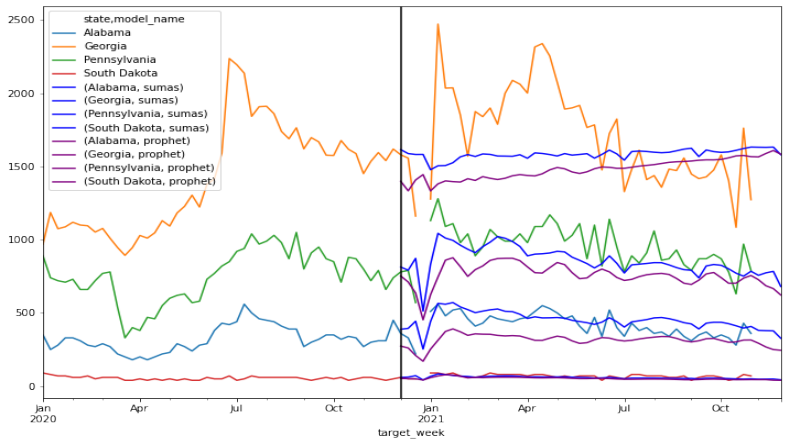
We obtained the weekly high-propensity business applications for all fifty states and the District of Columbia from the FRED economic database at the St. Louis Federal Reserve bank.[4] We use non-seasonally adjusted data.

This weekly business formation data in the United State is highly seasonal and has differing trends across states. This seasonality has many facets, which are driven by both time-year effects and by holidays that move from one year to the next. We forecast log data so that the data across states has more similar scales and transform it back to the original scale. We created a list of major American holidays, including them as features for our model (denoted Sumas), Prophet, DeepState, and AR.

For Sumas, Prophet, and DeepState, we censor the largest and smallest one percent changes. We also ran a univariate autoregressive model with lag-length chosen by Akaike's information criterion with the same holiday exogenous variables as in the other two models. Furthermore, we ran a VAR(2). The package we used does not support exogenous variables, and so we did not include any seasonal effects. Also, we are estimating a 51 variable VAR with no real regularization. It is primarily here as a standard model for comparison, but is not intended to be a true competitor.

We ran all the models every two months from January 2016 through November 2020. They have the same set of forecast dates. In the tables below, the error on the 4 week horizon is substantially higher than before or after it. This is likely due to error around December being substantially higher for all models. Since all models are using the exact same set of forecast dates, this should not affect the validity of the comparisons.

Figure 3: Business Formation Forecasts : 2020-11-30



As we can see in figure 3, Sumas is slightly more aggressive with its seasonal estimates than Prophet. In particular, it grows quicker following the drop around Christmas. However, like Prophet it is far smoother than the data, which being a forecast one would expect. We do not show the other models to avoid cluttering the graph.

[4]For example, the Alabama data is available at `https://fred.stlouisfed.org/series/HBUSAPPWNSAAL`.

For DeepState, we included the default weekly seasonality modeling and a trend, with LSTM cells, an initial burn-in period of 53 weeks (one year). We included the default 2 layers and 40 cells, but experimented with a few other specifications. We are reporting the best-performing one.

For Prophet, we included 10 Fourier terms and the holiday regressors mentioned previously. We did not include the default weekly seasonality. We used a linear trend and let the model automatically determine the change-points.

We compute mean absolute percentage error (MAPE) for each of the models by state, horizon. We use MAPE because the various states have different sizes – Californians start far more businesses than Wyomingites do.

In the mean comparison, we report statistical significance using the Diebold-Mariano test comparing each model to Sumas. If a model has higher (or lower) error than Sumas has and this difference is statistically significant at the 5% level, we add two stars. We also report the 5th and 95th percentile MAPEs across states and forecast dates to show which models are picking up the most signal and which models are the most robust.

Table 1: MAPE of Sumas, Prophet, and DeepState : Before February 2020

	Horizon	AR	DeepState	Prophet	Proposed	VAR
Mean	1	9.4**	13.9**	9.12**	10.6	11.7**
	2	9.3**	12.4**	9.5**	8.8	9.6**
	3	19.0**	24.6**	31.8	30.2	35.9**
	4	19.0**	20.9**	15.3	15.1	18.0**
	5	8.7**	16.4**	11.7**	12.6	14.1**
5th Percentile	1	0.5	1.1	0.5	0.8	0.9
	2	0.7	0.8	0.9	0.7	0.8
	3	1.1	1.1	0.8	1.0	1.3
	4	0.8	1.4	1.1	0.8	1.2
	5	0.6	1.2	0.9	1.1	1.2
95th Percentile	1	25.2	33.1	24.5	26.3	29.0
	2	25.1	31.2	26.2	23.4	25.2
	3	52.3	95.0	109.5	135.9	146.4
	4	77.3	67.3	46.9	45.3	45.3
	5	22.8	37.8	35.5	34.0	32.5

Before Covid-19, both univariate models (Prophet and AR) and Sumas all perform best some of the time. No one model is a clear winner. This is true both in the short- and long-horizons. Sumas does out-perform the multivariate models: DeepState and the VAR.

We then look at the Covid-19 and following periods only. As shown in figure 3, we see a clear change in the behavior of the data. This is when forecasting is difficult. You want a forecast that is fairly reactive to the new data, but not too reactive.

In short-horizons, Sumas is clearly the most accurate here, as shown in table 2. At the one week horizon, the mean error is approximately one-half of what Prophet and DeepState are producing, and about two-thirds of what AR, which was previously the best performing alternative were doing. In longer horizons, Sumas is usually the most accurate as well, but the AR model also does fairly well.

The out-performance of Sumas relative to Prophet and AR following Covid-19 is likely due to the partial pooling. Sumas regularizes the seasonal effects more and so is able to precisely estimate them even when the data are noisy. Its trend estimates appear to be about as accurate as the best performing models. This is why it continues to be competitive even over longer horizons.

We also see that Sumas is sufficiently flexible to fit patterns in the data. It has approximately half the 5th percentile error across the models except for the VAR at one week horizon. It is also rather robust. The 95th percentile error varies between one- and two-thirds of the other models errors at the one-week horizon.

Table 2: MAPE of Sumas, Prophet, and DeepState Post February 2020

	Horizon	AR	DeepState	Prophet	Proposed	VAR
Mean	1	13.1**	16.9**	18.2**	10.2	18.5**
	2	15.3**	19.5	17.1	18.1	19.4
	3	15.8**	18.9**	18.8**	12.4	21.7**
	4	15.7**	22.5**	21.8**	17.1	22.9**
	5	18.6**	21.6	20.4	21.1	21.1
5th Percentile	1	1.1	2.4	1.6	0.5	1.6
	2	1.6	1.4	1.4	1.0	1.8
	3	1.1	2.7	1.5	0.8	5.1
	4	0.7	1.7	1.1	1.0	2.3
	5	1.3	2.2	1.4	1.0	1.7
95th Percentile	1	33.3	40.1	47.4	25.6	53.4
	2	39.4	53.4	40.7	59.6	49.1
	3	43.6	41.6	41.9	30.5	45.0
	4	47.5	53.8	49.6	42.1	45.8
	5	44.4	49.6	46.8	50.1	50.4

5 Conclusion

We forecast series with large amount of variation driven by both static features, mainly seasonality, and dynamic changes in the level and growth rate. Furthermore, our data have medium-to-large time and cross-sectional dimensions, and we need to produce weekly forecasts. To achieve this we draw from the econometric and recurrent neural network literatures. In particular, we build a controllable, interpretable forecast using state-space techniques. This model scales to large cross-sections by partially pooling the structural parameters across the various nodes.

We develop a method that maps essentially arbitrary state-space models specified in terms of some human-interpretable structural parameters θ_n into some reduced form parameters $\Gamma(\theta_n, X_n)$. We then develop a method to partially pool these reduced-form parameters across the cross-section. Whenever the time-invariant part of the model drives a large portion of the variation, the model produces accurate interpretable forecasts even when some of the series are short and noisy.

Our model exploits the explainability of state-space models to separate the variation in the data into some underlying drivers and forecast each driver. This requires a conditionally linear propagation of shocks, but makes the forecast far easier to understand and control. As our customers are humans, this raises potential societal impacts. Because the model is not a black box and can be controlled, users of the model can use it to construct narratives that are not supported by the data thereby causing senior leaders to make decisions that are not well-founded by the evidence. As with any methodological system with no direct ethical issues, these models should only be used alongside methods for verifying results to catch any misleading narratives quickly.

We apply this model to forecasting business formation in the United States. We see that the model is competitive pre-Covid when the data are less noisy. It outperforms post-Covid-19 especially at shorter horizons, where it has between 1/2 and 2/3 the error of the state-of-the-art competitor models.

References

[1] Time series analysis by state space methods. `https://www.statsmodels.org/stable/statespace.html`. Accessed: 2021-02-02.

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[3] Jonathan R. Bradley, Scott H. Holan, and Christopher K. Wikle. Bayesian hierarchical models with conjugate full-conditional distributions for dependent data from the natural exponential family. *Journal of the American Statistical Association*, 115(532):2037–2052, 2020.

[4] Andrea Carriero, Todd E. Clark, and Massimiliano Marcellino. Large bayesian vector autoregressions with stochastic volatility and non-conjugate priors. *Journal of Econometrics*, 212(1):137–154, 2019. Big Data in Dynamic Predictive Econometric Modeling.

[5] Thomas Doan, Robert Litterman, and Christopher Sims. Forecasting and conditional projection using realistic prior distributions. *Econometric reviews*, 3(1):1–100, 1984.

[6] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*. Oxford university press, 2012.

[7] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[8] Christian Gourieroux, Alain Monfort, and Eric Renault. Indirect inference. *Journal of Applied Econometrics*, 8(S1):S85–S118, 1993.

[9] Gary Koop, Dimitris Korobilis, and Davide Pettenuzzo. Bayesian compressed vector autoregressions. *Journal of Econometrics*, 210(1):135–154, 2019. Annals Issue in Honor of John Geweke Complexity and Big Data in Economics and Finance: Recent Developments from a Bayesian Perspective.

[10] Ben O'Neill. Exchangeability, correlation, and bayes' effect. *International Statistical Review / Revue Internationale de Statistique*, 77(2):241–250, 2009.

[11] Giorgio E Primiceri. Time varying structural vector autoregressions and monetary policy. *The Review of Economic Studies*, 72(3):821–852, 2005.

[12] Syama Sundar Rangapuram, Matthias Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7796–7805, 2018.

[13] James H. Stock and Mark Watson. Dynamic factor models. In Michael P. Clements and David. Hendry, editors, *Oxford Handbook on Economic Forecasting*. Oxford University Press, 2011.

[14] S.J. Taylor and Lemtham B. Forecasting at scale. Technical report, Facebook, 2017.

[15] Feng Wang and David M.J. Tax. Survey on the attention based rnn model and its applications in computer vision. Technical report, arXiv, 2016.

[16] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *Procceedings of the 36th International Conference on Machine Learning*, pages 6607–6617, 2019.

[17] Roufeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Druv Madeka. A multi-horizon quantile recurrent forecaster. In *31st Conference on Nueral Information Processing Systems*, 2017.