Predictive State Recurrent Neural Networks

Carlton Downey
cmdowney@cs.cmu.edu

Ahmed Hefny
ahefny@cs.cmu.edu

Boyue Li
boyue@cs.cmu.edu

Byron Boots
bboots@cc.gatech.edu

Geoff Gordon
ggordon@cs.cmu.edu

Abstract: We present a new model, Predictive State Recurrent Neural Networks (PSRNNs), for filtering and prediction in dynamical systems. PSRNNs draw on insights from both Recurrent Neural Networks (RNNs) and Predictive State Representations (PSRs), and inherit advantages from both types of models. Like many successful RNN architectures, PSRNNs use (potentially deeply composed) bilinear transfer functions to combine information from multiple sources.

Keywords: Recurrent Neural Networks, Predictive State Representations

1 Introduction

Learning to predict temporal sequences of observations is a fundamental challenge in a range of disciplines including machine learning, robotics, and natural language processing. While there are a wide variety of different approaches to modelling time series data, many of these approaches can be categorized as either recursive Bayes Filtering or Recurrent Neural Networks.

Bayes Filters (BFs) [1] focus on modeling and maintaining a belief state: a set of statistics, which, if known at time t, are sufficient to predict all future observations as accurately as if we know the full history. The belief state is generally interpreted as the statistics of a distribution over the latent state of the data-generating process conditioned on history. BFs recursively update the belief state by conditioning on new observations using Bayes rule. Examples of common BFs include Hidden Markov Models (HMMs) [2] and Kalman Filters (KFs) [3].

Predictive State Representations [4] (PSRs) are a variation on Bayes filters that do not define system state explicitly, but proceed directly to a representation of state as the statistics of a distribution of features of *future observations*, conditioned on history. By defining the belief state in terms of observables rather than latent states, PSRs can be easier to learn than other filtering methods [5, 6, 7]. PSRs also support rich functional forms through kernel mean map embeddings [8], and a natural interpretation of model update behavior as a gating mechanism. This last is not unique to PSRs, as it is also possible to interpret the model updates of other BFs such as HMMs in terms of gating.

Due to their probabilistic grounding, BFs and PSRs possess a strong statistical theory leading to efficient learning algorithms. In particular, method-of-moments algorithms provide consistent parameter estimates for a range of BFs including PSRs [5, 7, 9, 10, 11]. Unfortunately, current versions of method of moments initialization restrict BFs to relatively simple functional forms such as linear-Gaussian (KFs) or linear-multinomial (HMMs).

Recurrent Neural Networks (RNNs) are an alternative to BFs that model sequential data via a parameterized internal state and update function. In contrast to BFs, RNNs are directly trained to minimize output prediction error, without adhering to any axiomatic probabilistic interpretation. Examples of popular RNN models include Long-Short Term Memory networks [12] (LSTMs), Gated Recurrent Units [13] (GRUs), and simple recurrent networks such as Elman networks [14].

RNNs and BFs offer complementary advantages and disadvantages: RNNs offer rich functional forms at the cost of statistical insight, while BFs possess a sophisticated statistical theory but are restricted to simpler functional forms in order to maintain tractable training and inference. By drawing insights from both Bayes Filters *and* RNNs, we develop a novel hybrid model, Predictive State Recurrent Neural Networks (PSRNNs). Like many successful RNN architectures, PSRNNs use (potentially

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

deeply composed) bilinear transfer functions to combine information from multiple sources. We show that such bilinear functions arise naturally from state updates in Bayes filters like PSRs, in which observations can be viewed as gating belief states. We show that PSRNNs directly generalize discrete PSRs, and can be learned effectively by combining Backpropogation Through Time (BPTT) with an approximately consistent method-of-moments initialization called two-stage regression (2SR). We also show that PSRNNs can be factorized using tensor decomposition, reducing model size and suggesting interesting connections to existing multiplicative architectures such as LSTMs. Finally, we note that our initialization for PSRNNs is consistent in the case of discrete data.

2 Background

2.1 Predictive State Representations

Predictive state representations (PSRs) [4] are a class of models for filtering, prediction, and simulation of discrete time dynamical systems. PSRs compactly represent state as a set of predictions of features of future observations. Specifically, a predictive state at time t is defined as $q_t = q_{t|t-1} = E[f_t \mid h_t]$, where $f_t = f(o_{t:t+k-1})$ is a vector of features of k future observations and $h_t = h(o_{1:t-1})$ is a vector of features of historical observations. The features are selected such that q_t determines the distribution of future observations $P(o_{t:t+k-1} \mid o_{1:t-1})$.[1] Filtering is the process of mapping a predictive state q_t to q_{t+1} conditioned on o_t, while prediction maps a predictive state $q_t = q_{t|t-1}$ to $q_{t+k|t-1} = E[f_{t+k} \mid o_{1:t-1}]$ without intervening observations.

By leveraging the recent concept of Hilbert Space embeddings of distributions [15], which can be used to embed the PSR in a Hilbert Space, we can generalize the PSR to continuous observations [8]. Hilbert Space Embeddings of PSRs (HSE-PSRs) [8] represent the state as one or more nonparametric conditional embedding operators in a Reproducing Kernel Hilbert Space (RKHS) [16] and use Kernel Bayes Rule (KBR) [15] to estimate, predict, and update the state. For a full treatment of HSE-PSRs see [8]. Let K_f, K_h, K_o be translation invariant kernels [17] defined on f_t, h_t, and o_t respectively. We use Random Fourier Features [17] (RFF) to define projections $\phi_t = RFF(f_t)$, $\eta_t = RFF(h_t)$, and $\omega_t = RFF(o_t)$ such that $K_f(f_i, f_j) = \phi_i^T \phi_j, \quad K_h(h_i, h_j) = \eta_i^T \eta_j, \quad K_o(o_i, o_j) = \omega_i^T \omega_j$. Using this notation, the HSE-PSR predictive state is $q_t = E[\phi_t \mid \eta_t]$. Formally an HSE-PSR (hereafter simply referred to as a PSR) consists of an initial state b_1, a 3-mode update tensor W, and a 3-mode normalization tensor Z. The PSR update equation is:

$$q_{t+1} = (W \times_3 q_t)\,(Z \times_3 q_t)^{-1} \times_2 o_t. \tag{1}$$

2.2 Two-stage Regression

Hefny et al. [7] show that PSRs can be learned by solving a sequence of regression problems. This approach, referred to as *Two-Stage Regression* or 2SR, is fast, statistically consistent, and reduces to simple linear algebra operations. In 2SR the PSR model parameters q_1, W, and Z are learned via the following set of equations:

$$q_1 = \frac{1}{T}\sum_{t=1}^{T} \phi_t, \quad W = \left(\sum_{t=1}^{T} \phi_{t+1} \otimes \omega_t \otimes \eta_t\right)\left(\sum_{t=1}^{T} \eta_t \otimes \phi_t\right)^{+}, \; Z = \left(\sum_{t=1}^{T} \omega_t \otimes \omega_t \otimes \eta_t\right)\left(\sum_{t=1}^{T} \eta_t \otimes \phi_t\right)^{+}.$$

where $\sum_{t=1}^{T} \phi_{t+1} \otimes \omega_t \otimes \eta_t$, $\sum_{t=1}^{T} \omega_t \otimes \omega_t \otimes \eta_t$ and $\sum_{t=1}^{T} \eta_t \otimes \phi_t$ are all estimated via regression. Here $+$ is the Moore-Penrose pseudo-inverse. We note that multiplying by the pseudo-inverse is also equivalent to solving an additional least squares regression problem, hence the name 2SR. Finally in practice we use ridge regression in order to improve model stability, and minimize the destabilizing effect of rare events while preserving consistency. Once we learn model parameters we train a regression model to predict ω_t from q_t.[2]

2.3 Tensor Decomposition

The tensor Canonical Polyadic decomposition (CP decomposition) [18] can be viewed as a generalization of the Singular Value Decomposition (SVD) to tensors. If $T \in \mathbb{R}^{(d_1 \times \ldots \times d_k)}$ is a tensor, then a

[1] For convenience we assume that the system is k-observable. At the cost of additional notation, this restriction could easily be lifted.

[2] Note that we can train a regression model to predict any quantity from the state. This is useful for general sequence-to-sequence mapping models. However, in this work we focus on predicting future observations.

CP decomposition of T is:

$$T = \sum_{i=1}^{m} a_i^1 \otimes a_i^2 \otimes ... \otimes a_i^k$$

where $a_i^j \in \mathbb{R}^{d_j}$ and $\otimes$ is the Kronecker product. The rank of T is the minimum m such that the above equality holds. In other words, the CP decomposition represents T as a sum of rank-1 tensors.

3 Predictive State Recurrent Neural Networks

We introduce Predictive State Recurrent Neural Networks (PSRNNs), a new RNN architecture inspired by PSRs. PSRNNs allow for a principled initialization via 2SR and refinement via BPTT. The key contributions which comprise PSRNNs are: 1) a new normalization scheme for PSRs which allows for effective refinement via BPTT; 2) the extension of the 2SR algorithm to a multilayered architecture; and 3) the optional use of a tensor decomposition to obtain a more scalable model.

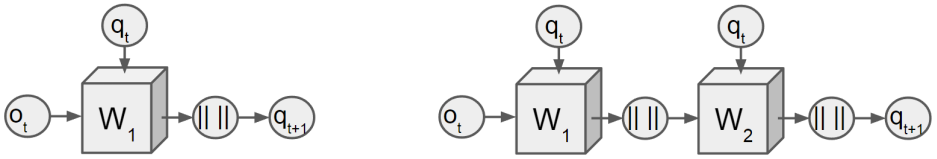
The basic building block of a PSRNN is a 3-mode tensor, which can be used to compute a bilinear combination of two input vectors. We note that, while bilinear operators are not a new development (e.g., they have been widely used in a variety of systems engineering and control applications for many years [19]), the current paper shows how to chain these bilinear components together into a powerful new predictive model.

Let q_t and o_t be the state and observation at time t. Let W be a 3-mode tensor, and let q be a vector. The 1-layer state update for a PSRNN is defined as:

$$q_{t+1} = \frac{W \times_2 o_t \times_3 q_t + b}{\|W \times_2 o_t \times_3 q_t + b\|_2} \tag{2}$$

Here the 3-mode tensor of weights W and the bias vector b are the model parameters.[3] This architecture is illustrated in Fig 1a. This model may appear simple, but crucially the tensor contraction $W \times_2 o_t \times_3 q_t$ integrates information from b_t and o_t multiplicatively, and acts as a gating mechanism.

To obtain a multilayer PSRNN, we stack the 1-layer blocks of Eq. (2) by providing the output of one layer as the observation for the next layer. (The state input for each layer remains the same.) In this way we can obtain arbitrarily deep RNNs. This architecture is displayed in Figure 1b.



(a) Single Layer PSRNN (b) Multilayer PSRNN

Figure 1: PSRNN architecture: See equation 2 for details. We omit bias terms to avoid clutter.

Finally we can use CP decomposition to factorize the model parameters in order to obtain a more scalable model. This model, called a *Factorized PSRNN*, is illustrated in figure 2.

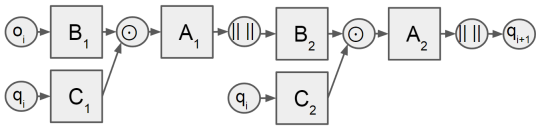


Figure 2: Factorized PSRNN Architecture

There are two components to learning PSRNNs: an initialization procedure followed by gradient-based refinement.

[3] We define $A \times_p B$ to the be contraction of A and B along the pth mode, e.g., $[W \times_2 q]_{i,j} = \sum_k W_{i,l,j} q_l$.

We use the two-stage regression algorithm of Hefny et al. [7] described above to initialize 1-layer PSRNNs: It is not immediately clear how to extend this approach to multilayered PSRNNs; but, suppose we have learned a 1-layer PSRNN P using two-stage regression. We can use P to perform filtering on a dataset to generate a sequence of estimated states $\hat{q}_1, ..., \hat{q}_n$. According to the architecture described in Figure 1b, these states are treated as observations in the second layer. Therefore we can initialize the second layer by an additional iteration of two-stage regression *using our estimated states $\hat{q}_1, ..., \hat{q}_n$ in place of observations*. This process can be repeated as many times as desired to initialize an arbitrarily deep PSRNN.

Once we have obtained a PSRNN using the 2SR approach described above, we can use BPTT to refine the PSRNN. We note that one of the reasons we choose to use 2-norm divisive normalization is that it is not practical to perform BPTT through the matrix inverse required in PSRs. We analyze learning in more detail in full version of the paper, including consistency and connection to gating.

4 Experiments

We use the following datasets in our experiments:

- **Swimmer** We consider the 3-link simulated swimmer robot from the open-source package OpenAI gym.[4] The observation model returns the angular position of the nose as well as the angles of the two joints. We collect 25 trajectories from a robot that is trained to swim forward (via the cross entropy with a linear policy), with a train/test split of 20/5.
- **Mocap** This is a Human Motion Capture dataset consisting of 48 skeletal tracks from three human subjects collected while they were walking. The tracks have 300 timesteps each, and are from a Vicon motion capture system. We use a train/test split of 40/8. Features consist of the 3D positions of the skeletal parts (e.g., upper back, thorax, clavicle).
- **Handwriting** This is a digit database available on the UCI repository [20, 21] created using a pressure sensitive tablet and a cordless stylus. Features are x and y tablet coordinates and pressure levels of the pen at a sampling rate of 100 milliseconds. We use 25 trajectories with a train/test split of 20/5.

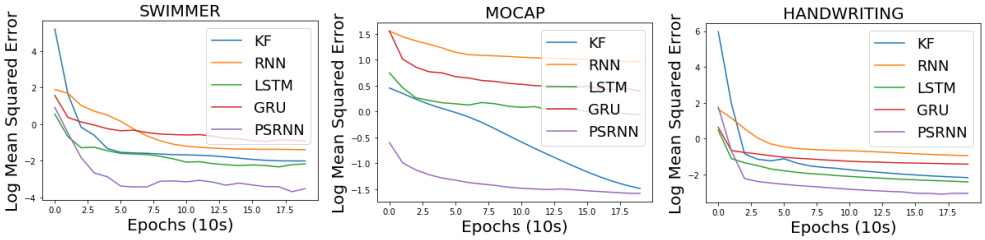


Figure 3: Validation MSE vs Epoch on the Swimmer, Mocap, and Handwriting datasets

In Figure 3 we compare model performance on the Swimmer, Mocap, and Handwriting datasets. We see that PSRNNs significantly outperform alternative approaches on all datasets.

5 Conclusions

We present PSRNNs: a new approach for modelling time-series data that hybridizes PSRs and RNNs. PSRNNs have both a principled initialization procedure and a rich functional form. The basic PSRNN block consists of a 3-mode tensor, corresponding to bilinear combination of the state and observation, followed by divisive normalization. These blocks can be arranged in layers to increase the expressive power of the model. Tensor CP decomposition can be used to obtain factorized PSRNNs, which allow flexibly selecting the number of states and model parameters. We applied PSRNNs to 3 datasets and showed that we outperform alternative approaches in all cases.

[4] https://gym.openai.com/

References

[1] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Comput.*, 11(2):305–345, Feb. 1999. ISSN 0899-7667. doi:10.1162/089976699300016674. URL http://dx.doi.org/10.1162/089976699300016674.

[2] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[3] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

[4] M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *In Advances In Neural Information Processing Systems 14*, pages 1555–1561. MIT Press, 2001.

[5] B. Boots, S. Siddiqi, and G. Gordon. Closing the learning planning loop with predictive state representations. *International Journal of Robotics Research (IJRR)*, 30:954–956, 2011.

[6] B. Boots and G. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)*, 2011.

[7] A. Hefny, C. Downey, and G. J. Gordon. Supervised learning for dynamical system learning. In *Advances in Neural Information Processing Systems*, pages 1963–1971, 2015.

[8] B. Boots, G. J. Gordon, and A. Gretton. Hilbert space embeddings of predictive state representations. *CoRR*, abs/1309.6819, 2013. URL http://arxiv.org/abs/1309.6819.

[9] D. J. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. *CoRR*, abs/0811.4413, 2008.

[10] A. Shaban, M. Farajtabar, B. Xie, L. Song, and B. Boots. Learning latent variable models by improving spectral solutions with exterior point methods. In *Proceedings of The International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2015.

[11] P. Van Overschee and B. De Moor. N4sid: numerical algorithms for state space subspace system identification. In *Proc. of the World Congress of the International Federation of Automatic Control, IFAC*, volume 7, pages 361–364, 1993.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi:10.1162/neco.1997.9.8.1735.

[13] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[14] J. L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[15] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[16] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[17] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[18] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189, 1927.

[19] L. Ljung. *System identification*. Wiley Online Library, 1999.

[20] F. A. E. Alpaydin. Pen-Based Recognition of Handwritten Digits Data Set. https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.

[21] E. Alpaydin and F. Alimoglu. Pen-based recognition of handwritten digits data set. *University of California, Irvine, Machine Learning Repository. Irvine: University of California*, 1998.