Factored Temporal Sigmoid Belief Networks for Sequence Learning

Jiaming Song[1] Zhe Gan[2] and Lawrence Carin[2]

[1]Department of Computer Science and Technology
Tsinghua University

[2]Department of Electrical and Computer Engineering
Duke University

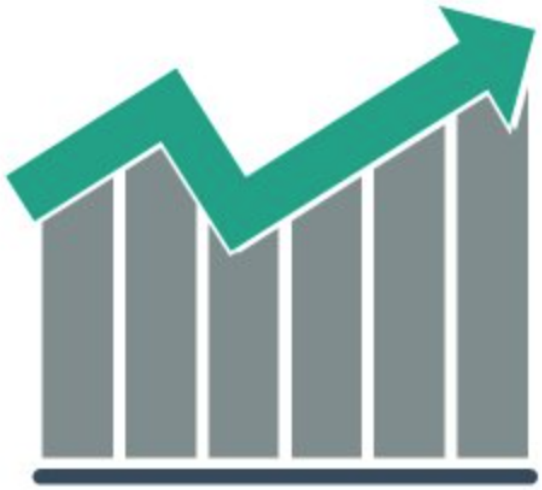
June 21, 2016

Outline

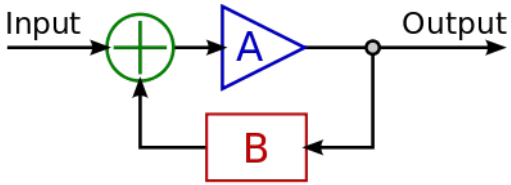
Background

Time series analysis has wide applications

Weather Forecast



Quantitative Finance



Control Engineering

Sequence Modeling in Deep Learning

Deep Neural Networks - without latent variables

- Recurrent Neural Networks (RNN)
- Long Short Term Memory (LSTM)

Deep Generative Models - with latent variables

Built upon non-temporal models, such as

- Restricted Boltzmann Machine (RBM)
- Sigmoid Belief Network (SBN)
- Variational AutoEncoder (VAE)

Deep Generative Models for Sequence Modeling

Original Model	$\Longrightarrow$	Temporal Model
RBM	$\Longrightarrow$	**Temporal** RBM[1], **Recurrent Temporal** RBM[2]
SBN	$\Longrightarrow$	**Temporal** SBN[3]
VAE	$\Longrightarrow$	Variational **Recurrent** AE[4]

[1] Sutskever and Hinton, *Learning Multilevel Distributed Representations for High-Dimensional Sequences*.

[2] Sutskever et al., *The Recurrent Temporal Restricted Boltzmann Machine*

[3] Gan et al., *Deep Temporal Sigmoid Belief Networks for Sequence Modeling*.

[4] Fabius and Amersfoort, *Variational Recurrent Auto-Encoders*.

Problem

What if we want to

1. generate **mulitple styles** of sequences from a **single** model?
2. **control** the style of sequence during generation?
3. **combine** multiple styles to generate a new style?

Need side information $\boldsymbol{y}$ to distinguish styles + **a conditional model**.

SBN vs. RBM

Base Model	SBN	RBM
Temporal Model	Temporal SBN	TRBM / RTRBM
Conditional Model	Conditional TSBN	Conditional RBM
Factored Model	Factored CTSBN	Factored CRBM

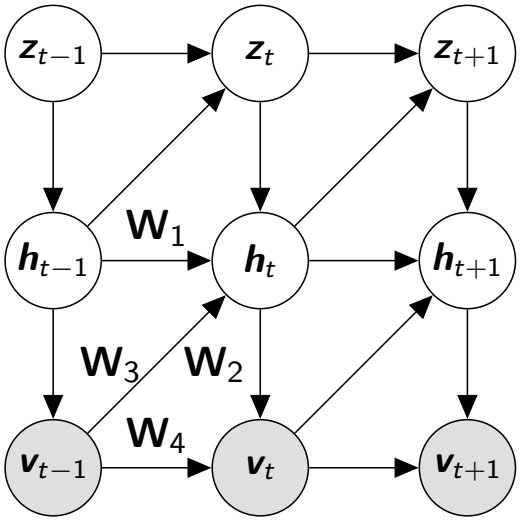
SBN vs. RBM

Base Model	SBN	RBM
Temporal Model	Temporal SBN	TRBM / RTRBM
Conditional Model	Conditional TSBN	Conditional RBM
Factored Model	Factored CTSBN	Factored CRBM

Outline

Temporal Sigmoid Belief Network (TSBN)

A deep **directed** generative model for modeling **discrete** time series data.



v_t observation at time t.

h_t, z_t latent variables at time t.

W_i model parameters.

Formulation of TSBN

Single layer joint probability:

$$p_\theta(\mathbf{V}, \mathbf{H}) = p(\boldsymbol{h}_1)p(\boldsymbol{v}_1|\boldsymbol{h}_1)\prod_{t=2}^{T} p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}, \boldsymbol{v}_{t-1}) \cdot p(\boldsymbol{v}_t|\boldsymbol{h}_t, \boldsymbol{v}_{t-1}) \quad (1)$$

where

$$p(h_{jt} = 1|\boldsymbol{h}_{t-1}, \boldsymbol{v}_{t-1}) = \sigma(\mathbf{W}_1\boldsymbol{h}_{t-1} + \mathbf{W}_3\boldsymbol{v}_{t-1} + \boldsymbol{b})$$
$$p(v_{jt} = 1|\boldsymbol{h}_t, \boldsymbol{v}_{t-1}) = \sigma(\mathbf{W}_2\boldsymbol{h}_t + \mathbf{W}_4\boldsymbol{v}_{t-1} + \boldsymbol{b})$$

Side Information

Provide additional side information $\boldsymbol{y}_t$ at each frame t during generation:

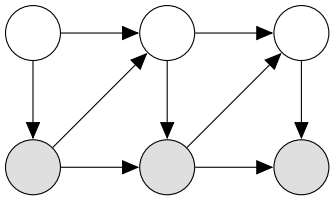
$$p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}, \boldsymbol{v}_{t-1}) \Longrightarrow p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}, \boldsymbol{v}_{t-1}, \boldsymbol{y}_t)$$
$$p(\boldsymbol{v}_t|\boldsymbol{h}_t, \boldsymbol{v}_{t-1}) \Longrightarrow p(\boldsymbol{v}_t|\boldsymbol{h}_t, \boldsymbol{v}_{t-1}, \boldsymbol{y}_t)$$

We assume that $\boldsymbol{y}_t$ is a vector with S elements.

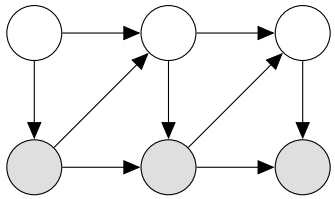
Conditional Generation

Consider the case where $\boldsymbol{y}_t$ is a one-hot encoded vector for styles.

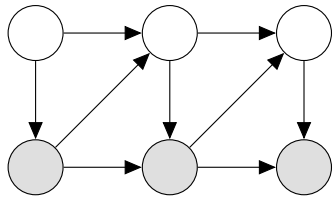
$[1, 0, 0, \cdots]$
Walking



$[0, 1, 0, \cdots]$
Running



$[0, 0, 1, \cdots]$
Dancing



Train **one model** for **each style**, and combine the models.

Conditional TSBN

Generalize that to real valued $\boldsymbol{y}_t$:

$$\tilde{\boldsymbol{h}}_t = \mathbf{W}_1^{(y)} \boldsymbol{h}_{t-1} + \mathbf{W}_3^{(y)} \boldsymbol{v}_{t-1} + \boldsymbol{b}^{(y)} \tag{2}$$

$$\tilde{\boldsymbol{v}}_t = \mathbf{W}_2^{(y)} \boldsymbol{h}_t + \mathbf{W}_4^{(y)} \boldsymbol{v}_{t-1} + \boldsymbol{c}^{(y)} \tag{3}$$

where $\boldsymbol{b}^{(y)} = \mathbf{B}\boldsymbol{y}_t$, $\boldsymbol{c}^{(y)} = \mathbf{C}\boldsymbol{y}_t$, and $\mathbf{W}_{i(jk)}^{(y)} = \sum_{s=1}^{S} \hat{\mathbf{W}}_{i(jks)} y_{st}$.

The model parameter $\hat{\mathbf{W}}_i$ is a three way tensor.

Problems:

1. Too many params;
2. Poor generalization.

Factoring Weight Parameters

We factor the weight matrices $\mathbf{W}^{(y)} \in \mathbb{R}^{J \times M}$ as

$$\mathbf{W}^{(y)} = \mathbf{W}_a \cdot \text{diag}(\mathbf{W}_b \boldsymbol{y}_t) \cdot \mathbf{W}_c \tag{4}$$

$\mathbf{W}_a \in \mathbb{R}^{J \times F}$, $\mathbf{W}_b \in \mathbb{R}^{F \times S}$ and $\mathbf{W}_c \in \mathbb{R}^{F \times M}$. F is the number of factors.

$\mathbf{W}_a$ **input**-to-**factor** relationship;

$\mathbf{W}_c$ **factor**-to-**output** relationship;

$\text{diag}(\mathbf{W}_b \boldsymbol{y}_t)$ **factor**-to-**factor** relationship for each style.

Factoring Weight Parameters (cont'd)

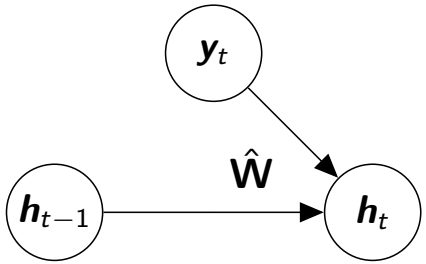


Figure: Non-factored weights

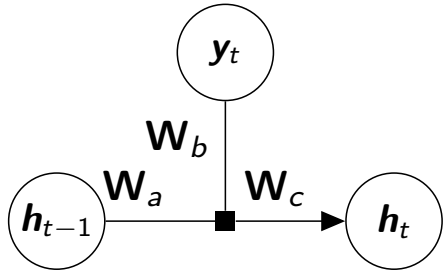


Figure: Factored weights

Advantages of factoring:

1. Reduces the number of parameters from $J \cdot M \cdot S$ to $(J + M + S) \cdot F$.
2. Have parameters that **explicitly** capture the similarties among styles ($\mathbf{W}_a$ and $\mathbf{W}_c$).

Deep Architecture

For a network with L layers:

$$p(\boldsymbol{h}_t^{(L)}) = \prod_{j=1}^{J^{(L)}} p(h_{jt}^{(L)} | \boldsymbol{h}_{t-1}^{(L)}, \boldsymbol{h}_{t-1}^{(L-1)}, \boldsymbol{y}_t) \tag{5}$$

$$\cdots$$

$$p(\boldsymbol{h}_t^{(\ell)}) = \prod_{j=1}^{J^{(\ell)}} p(h_{jt}^{(\ell)} | \boldsymbol{h}_t^{(\ell+1)}, \boldsymbol{h}_{t-1}^{(\ell)}, \boldsymbol{h}_{t-1}^{(\ell-1)}, \boldsymbol{y}_t) \tag{6}$$

$$\cdots$$

$$p(\boldsymbol{h}_t^{(1)}) = \prod_{j=1}^{J^{(1)}} p(h_{jt}^{(1)} | \boldsymbol{h}_t^{(2)}, \boldsymbol{h}_{t-1}^{(1)}, \boldsymbol{v}_{t-1}, \boldsymbol{y}_t) \tag{7}$$

Outline

Scalable Learning and Inference

A recognition model $q_\phi(\mathbf{H}|\mathbf{V},\mathbf{Y})$ to approximate the posterior $p(\mathbf{H}|\mathbf{V},\mathbf{Y})$, with the following objective:

$$\mathcal{L}(\mathbf{V}|\mathbf{Y},\theta,\phi) = \mathbb{E}_q[\log q_\phi(\mathbf{H}|\mathbf{V},\mathbf{Y}) - \log p_\theta(\mathbf{V},\mathbf{H}|\mathbf{Y})] \tag{8}$$

Recognition Model q_ϕ

$$q_\phi(\mathbf{H}|\mathbf{V},\mathbf{Y}) = \prod_{t=1}^{T} q(\boldsymbol{h}_t|\boldsymbol{h}_{t-1},\boldsymbol{v}_t,\boldsymbol{v}_{t-1},\boldsymbol{y}_t) \tag{9}$$

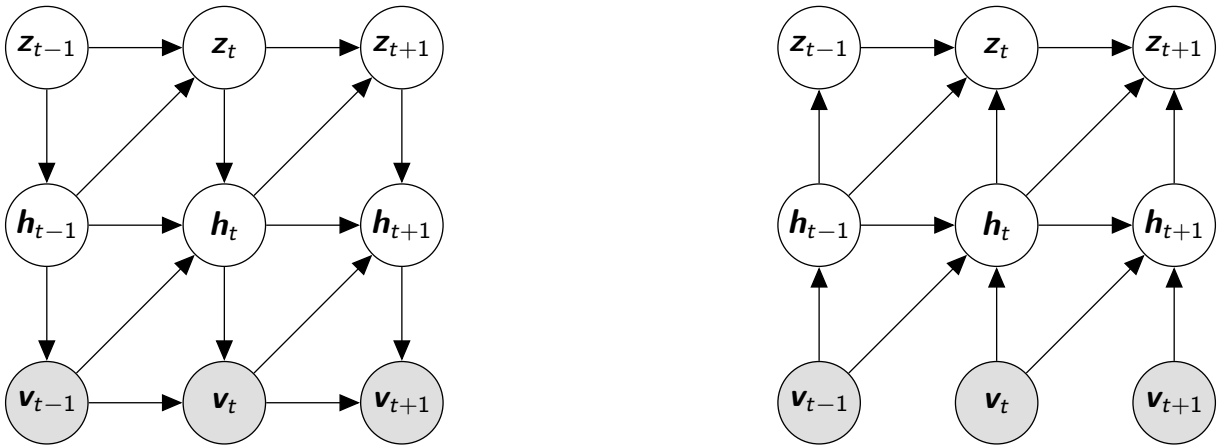


Figure: Generative model (left) and recognition model (right).

Semi-supervised Learning

Obtaining labels for sequential data might be expensive (e.g. documents).

Semi-supervised framework can:

- Train a generative model and a classifier;
- Make use of unlabeled data.

Generative model:

$$p_\theta(\mathbf{V}, \mathbf{H}, \mathbf{Y}) = p_\theta(\mathbf{Y}; \pi) \cdot p_\theta(\mathbf{V}, \mathbf{H}|\mathbf{Y}) \tag{10}$$

where $p_\theta(\mathbf{Y}; \pi)$ is the prior distribution of $\mathbf{Y}$.

Semi-supervised Learning (cont'd)

A recognition model for both $\mathbf{H}$ and $\mathbf{Y}$:

$$q_\phi(\mathbf{H}, \mathbf{Y}|\mathbf{V}) = q_\phi(\mathbf{H}|\mathbf{V}, \mathbf{Y}) \cdot q_\phi(\mathbf{Y}|\mathbf{V}) \tag{11}$$

$q_\phi(\mathbf{Y}|\mathbf{V})$ denotes the classifier.

The objective function contains:

1. Labeled data:
   - Generative loss: $\mathbb{E}_q[\log q_\phi(\mathbf{H}|\mathbf{V}, \mathbf{Y}) - \log p_\theta(\mathbf{V}, \mathbf{H}|\mathbf{Y})]$
   - Discriminative loss: $\mathbb{E}_{\tilde{p}_l(\mathbf{V},\mathbf{Y})}[\log q_\theta(\mathbf{Y}|\mathbf{V})]$
2. Unlabeled data: $\mathbb{E}_q[\log q_\phi(\mathbf{H}, \mathbf{Y}|\mathbf{V}) - \log p_\theta(\mathbf{H}, \mathbf{V}, \mathbf{Y})]$

Outline

Tasks & Models

Prediction Given $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1}$, predict $\boldsymbol{v}_t$.

Generation Given $\boldsymbol{v}_1, \ldots \boldsymbol{v}_t$, generate $\boldsymbol{v}_t, \boldsymbol{v}_{t+1} \ldots$.

Classification Given $\boldsymbol{v}_1, \ldots \boldsymbol{v}_n$, identify the style $\boldsymbol{y}_n$.

CTSBN Conditional Temporal SBN.

FCTSBN Factored Conditional TSBN.

dFCTSBN Two-layer Factored CTSBN.

mocap2 Prediction

Motion capture data with **2 styles** (walking and running).

Method	Walking	Running
FCTSBN	**4.59** $\pm$ 0.35	**2.86** $\pm$ 0.23
CTSBN	4.67 $\pm$ 0.22	3.41 $\pm$ 0.65
TSBN	5.12 $\pm$ 0.50	4.85 $\pm$ 1.26
dFCTSBN	**4.31** $\pm$ 0.13	2.58 $\pm$ 0.21
DTSBN-S	4.40 $\pm$ 0.28	**2.56** $\pm$ 0.40
DTSBN-D	4.62 $\pm$ 0.01	2.84 $\pm$ 0.01
ss-SRTRBM	8.13 $\pm$ 0.06	5.88 $\pm$ 0.05
g-RTRBM	14.41 $\pm$ 0.38	10.91 $\pm$ 0.27

Table: Prediction error obtained for the mocap2 dataset.

mocap10 Generation

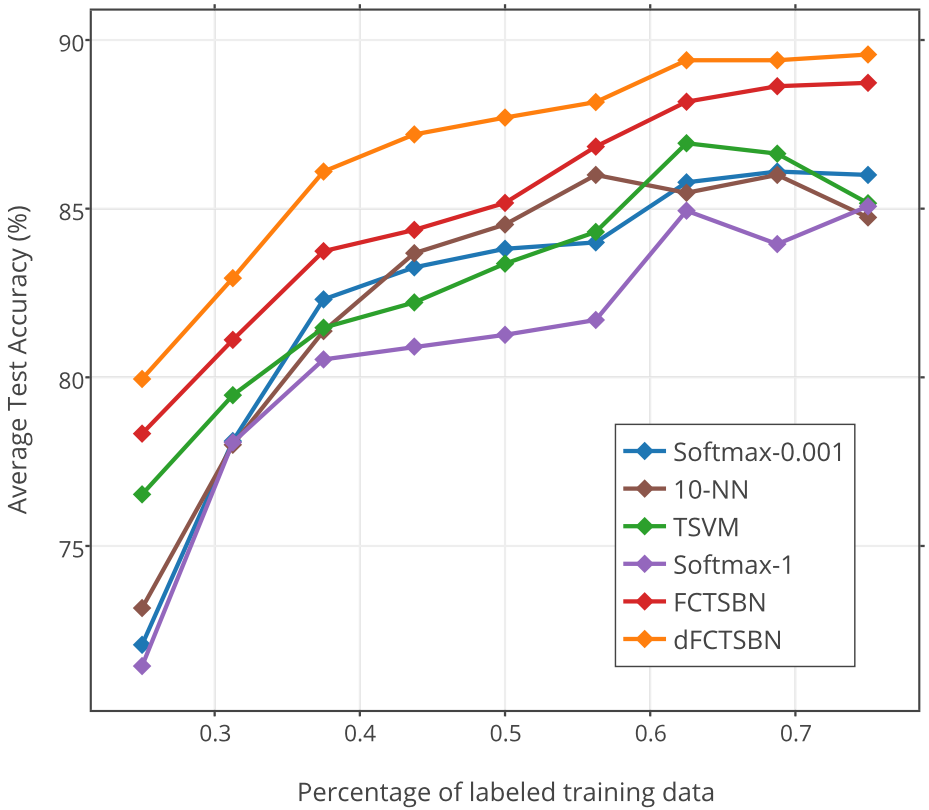
Motion capture data with **10 styles**, but with less frames in each style.

We can generate 9 styles, with style transitions and combinations.

chicken to **sexy**.

strong to **drunk**.

mocap10 Semi-supervised Learning (Classifier)



Baselines

NN Nearest Neighbors

TSVM Transductive SVM

Softmax Softmax classifier with L2 regularization.

mocap2 Semi-supervised Learning (Generator)

- 33 motion videos in total.
- Only 1 video for each style is labeled.

jog to **walk**

Weather Prediction

Question

How does the representation of side information affect performance?

Weather data from 25 different locations, on a 5×5 **grid**.

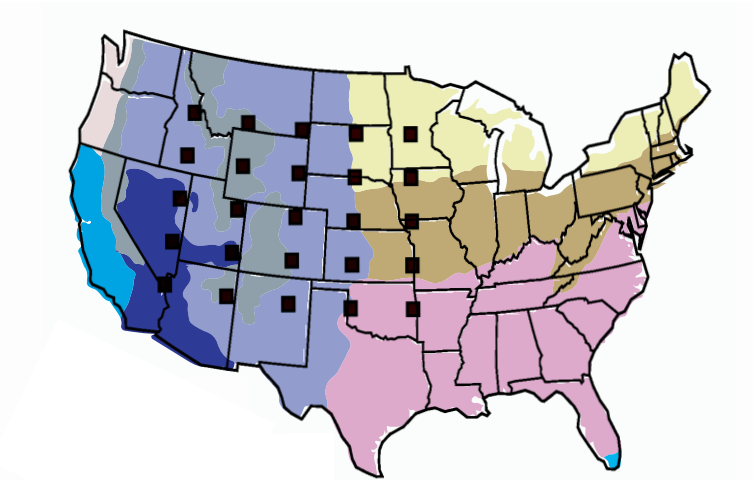


Figure: Areas with colors indicate climate zones.

Weather Prediction (cont'd)

Three types of side information:

$[-1.41, 1.41]$	$[0, 0, 1, 0, 0; 1, 0, 0, 0, 0]$	$[0, \ldots, 0, 1, 0, \ldots, 0]$
Geo-location (2d)	Concatenated vectors, for grid position (10d)	One-hot encoded (25d)

Weather Prediction (cont'd)

	2d	10d	25d
CTSBN	6.45 ± 0.11	3.83 ± 0.20	3.78 ± 0.01
FCTSBN	5.46 ± 0.06	3.43 ± 0.03	3.37 ± 0.02
dFCTSBN	$\mathbf{5.09} \pm 0.11$	$\mathbf{3.37} \pm 0.01$	$\mathbf{3.35} \pm 0.02$
FCRBM	5.62 ± 0.35	3.77 ± 0.38	3.75 ± 0.08

Table: Average prediction error of 25 locations on the weather dataset.

- The 25d CTSBN is **12.5x larger** than 2d version.
- The 25d FCTSBN is **less than 1.25x larger**
  - much more scalable style-wise!

Outline

Takeaway

- A deep directed generative model for multiple sequences with styles.
- Additional side information allows for conditional generation.
- Factoring reduces the model complexity while improving generalization.
- The generative model is a good regularizer for semi-supervised learning.