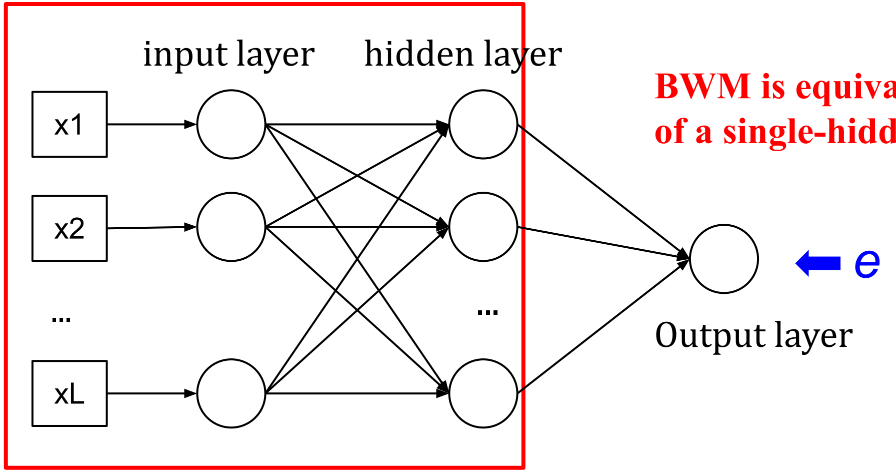
Training a Bank of Wiener Models with a Novel Quadratic Mutual Information Cost Function

Bo Hu and Dr. Jose C. Principe (Life Fellow, IEEE)
Dept. of Electrical and Computer Engineering, University of Florida

This work was partially supported by NSF award 1747783.
We would like to thank NSF for sponsoring this research.

Problems of time-delay neural networks (TDNN)



BWM is equivalent to the first layer of a single-hidden layer TDNN

To train the TDNN, error signals (MSE) are formed in the final layer. Then the parameters are trained by BP.

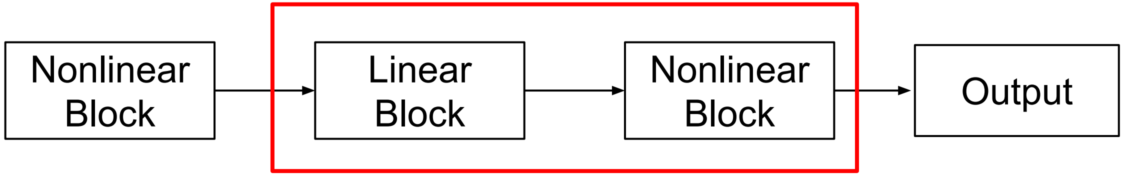
- **Backpropagation**: Each unit will only receive the gradient information from the top layer.

- **Spurious Correlation**: The output layer of the MLP creates spurious correlations between the units.

- **MSE**: It's only an error measurement based on the second moments. It can not be used if the dimensionalities mismatch.

1. Bank of Wiener models

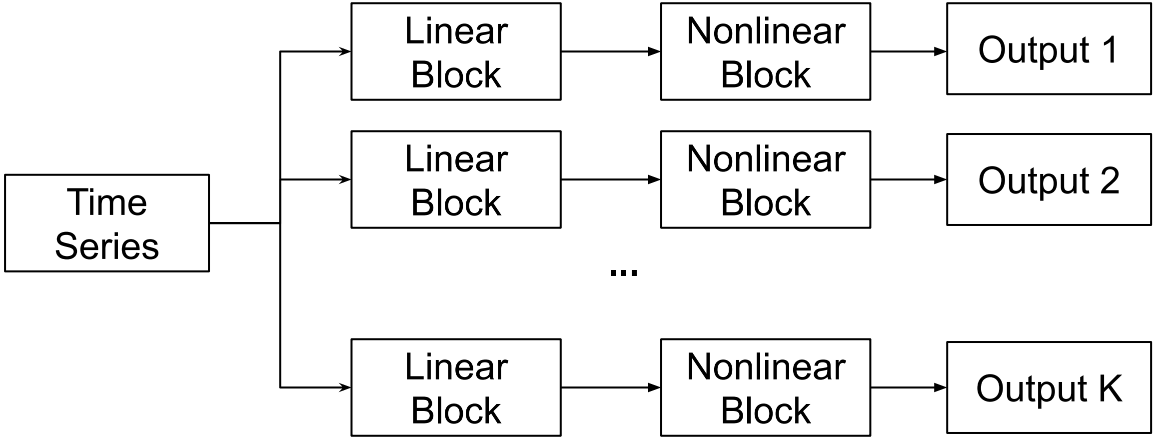
- **Block-oriented models**



Wiener model

SISO system

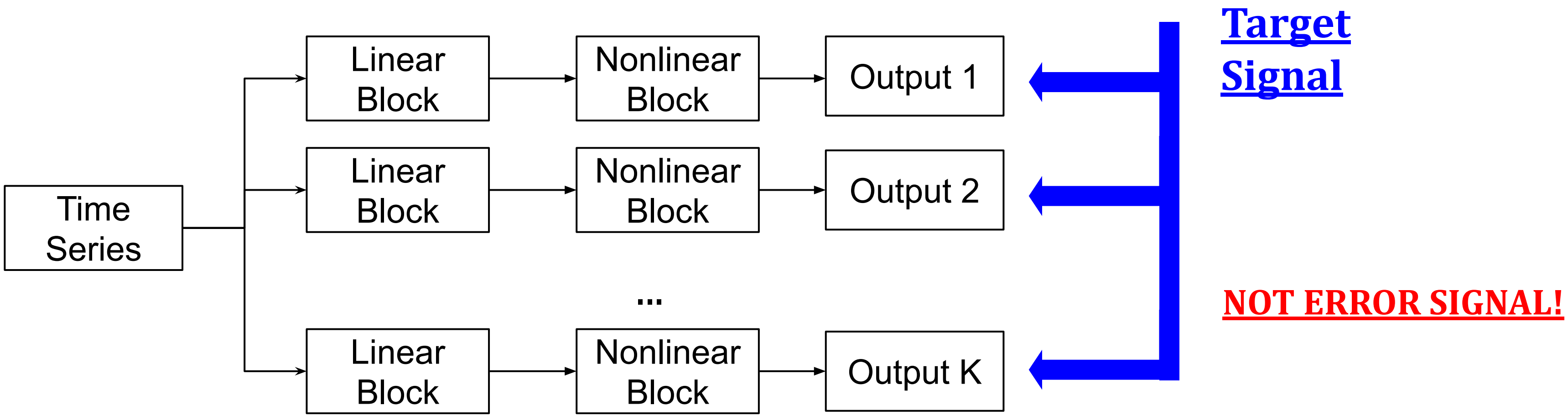
- **Bank of Wiener models** (BWM)



SIMO/MIMO system

- BWM forms a SIMO/MIMO system. BWM outputs learns explicitly the bases of a **projection space** by bringing the desired as the target to the hidden layer.

- BWM has the same structure as the first layer of a single-hidden-layer TDNN. Training BWM *does not* require **backpropagation** (BP). There is no spurious correlations from a two-layer neural networks.



Given a BWM with K Wiener models. For the k-th Wiener model.

The k-th output: $h_{\theta_k}(\mathbf{x}) := \mathbf{z}_{\theta_k} = \sigma(\mathbf{w}_k^{\mathsf{T}}\mathbf{x} + b_k)$

The series of Wiener models outputs: $\mathbf{z}_{\theta_{1:K}} = [\mathbf{z}_{\theta_1}, \mathbf{z}_{\theta_2} \ldots \mathbf{z}_{\theta_K}]^{\mathsf{T}}$

The linear combination of BWM outputs defines a **projection space**. After training BWM, a least-square solution is applied to yield the minimum MSE.

To train this model, we have to use a **new cost function** → **quadratic mutual information (QMI)**

2. Quadratic mutual information

Rényi's entropy: $H_\alpha(p) = \frac{1}{1-\alpha}\log_2(\sum_{x\in\mathcal{X}} p^\alpha(x))$

$$H_2(p) = -\log_2(\sum_{x\in\mathcal{X}} p^2(x))$$

Gaussian kernel: $k_\sigma(\mathbf{x}_i - \mathbf{x}_j) = \frac{1}{(2\pi)^{p/2}\cdot\sigma^p}\exp(-\frac{1}{2\sigma^2}\cdot\|\mathbf{x}_i - \mathbf{x}_j\|_2^2)$

Parzen density estimator: $\tilde{p}(x) = \frac{1}{N}\sum_{n=1}^{N} k_\sigma(x - x_n)$

The estimator to Rényi's entropy becomes

$$\begin{aligned} V_E(\mathbf{X}) &= -\log_2(\sum_{x\in\mathcal{X}} \tilde{p}^2(x)) \\ &= -\log_2\left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} k_\sigma(x - x_i)k_\sigma(x - x_j)\right) \\ &= -\log_2\left(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} k_\sigma(x_i - x_j)\right) \end{aligned}$$

Similarly, given sample pairs $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=0}^{N}$, we can estimate the joint entropy as

$$V_J(\mathbf{X}, \mathbf{Y}) = -\log_2(\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} k_\sigma(x_i, x_j) \cdot k_\sigma(y_i, y_j))$$

The **quadratic mutual information** (QMI) can be constructed in the form

$$I_Q(\mathbf{X}, \mathbf{Y}) = V_E(\mathbf{X}) + V_E(\mathbf{Y}) - V_J(\mathbf{X}, \mathbf{Y})$$

QMI has been broadly used in machine learning and time series analysis:

- Jose C. Principe. Information theoretic learning: Renyi'sentropy and kernel perspectives, 2010.
- Dongxin Xu and Jose C. Principe. Training mlps layer-by-layer with the information potential, IJCNN, 1999.
- Luis G. Sanchez Giraldo and Jose C. Principe. Information theoretic learning with infinitely divisible kernels, ICLR, 2013
- Austin J. Brockmeier, John S. Choi, Evan G. Kriminger, Joseph T Francis, and Jose C. Principe. Neural decoding with kernel-based metric learning, Neural Computation, 2014

We can easily write the density estimation in the expectation form:

$$\tilde{p}(x) = \mathbb{E}_{\mathbf{x}' \sim P}[k_\sigma(x - \mathbf{x}')]$$

Similarly, we can use this form to estimate Rényi's entropy of a given distribution:

$$\nu_E(\mathbb{P}) = -\log_2\left(\sum_{x \in \mathcal{X}} \tilde{p}(x)p(x)\right)$$
$$= -\log_2(\mathbb{E}_{\mathbf{x} \sim P, \mathbf{x}' \sim P}[k_\sigma(\mathbf{x} - \mathbf{x}')])$$

$$\nu_J(\mathbb{P}_{XY}) = -\log_2\left(\mathbb{E}[k_\sigma(x - x') \cdot k_\sigma(y - y')]\right)$$

The equivalent form for quadratic mutual information is thus

$$I_{EQ}(\mathbb{P}_{XY}) = \nu_E(\mathbb{P}_X) + \nu_E(\mathbb{P}_Y) - \nu_J(\mathbb{P}_{XY})$$

We call this form the **empirical embedding of QMI** (E-QMI).

This cost function can be used to quantify the dependency across different dimensions, thus fits perfectly for training BWM

4. Designing QMI cost functions to train BWM

Why are modifications needed to use E-QMI as a cost functions?

marginal pdf

$$I_{EQ}(\mathbb{P}_{\{f(\mathbf{x}),\mathbf{d}\}}) = -\log_2\{\frac{\mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))] \cdot \mathbb{E}[k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)]}{\mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2)) \cdot k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)]}\}$$

joint pdf

By Cauchy-Schwarz inequality, we have

$$\mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))] \cdot \mathbb{E}[k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)] \leq \mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2)) \cdot k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)]$$

Thus we always have $I_{EQ}(\mathbb{P}_{\{f(\mathbf{x}),\mathbf{d}\}}) \geq 0$ for any mapper.

However if we maximize the cost function, it could be unbounded since now we have a **SIMO/MIMO** system.

Type-I normalization

Normalize the model output and the target by their standard deviations

In this way, we keep each Wiener model output and the target in the proper range.

The BWM outputs: $\mathbf{z}'_{\theta_{1:K}} = [\frac{\mathbf{z}_{\theta_1}}{\text{std}[\mathbf{z}_{\theta_1}]}, \frac{\mathbf{z}_{\theta_2}}{\text{std}[\mathbf{z}_{\theta_2}]} \cdots \frac{\mathbf{z}_{\theta_K}}{\text{std}[\mathbf{z}_{\theta_K}]}]^\intercal$

The target: $\mathbf{d}' = \frac{\mathbf{d}}{\text{std}[\mathbf{d}]}$

Type-I cost: $\underset{\{\theta_1,\theta_2\ldots\theta_K\}}{\text{maximize}} \quad I_{EQ}(\mathbb{P}_{\{\mathbf{z}'_{\theta_{1:K}},\mathbf{d}'\}})$

Type-II normalization

Suppose the special case $f(\mathbf{x}) = \mathbf{d}$, the value of the cost function becomes

$$I_{EQ}(\mathbb{P}_{\{f(\mathbf{x}),\mathbf{d}\}} = \mathbb{P}_{\{f(\mathbf{x}),f(\mathbf{x})\}}) = -\log_2\{\frac{\mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))]^2}{\mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))^2]}\}$$

By adding a constant, we want this term to satisfy

$$2 \cdot \mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2)) + b]^2 = \mathbb{E}[(k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2)) + b)^2]$$

Solving for b, we obtain the solution

$$b = \mathrm{std}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))] - \mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))]$$

Using this normalization scheme, we can construct

$$\begin{aligned}
\mathbf{z}_f &= k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2)) - \mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))] + \mathrm{std}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))] \\
\mathbf{z}_d &= k_\sigma(\mathbf{d}_1 - \mathbf{d}_2) - \mathbb{E}[k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)] + \mathrm{std}[k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)] \\
i_{EQ}(\mathbb{P}_{\{f(\mathbf{x}),\mathbf{d}\}}) &= -\log_2\left[\mathbb{E}[\mathbf{z}_f] \cdot \mathbb{E}[\mathbf{z}_d]\right] + \log_2\left[\mathbb{E}[\mathbf{z}_f \cdot \mathbf{z}_d]\right]
\end{aligned}$$

which will always be bounded by 1.

To train the BWM, we take the summation of all Wiener model outputs

$$\underset{\{\theta_1,\theta_2\ldots\theta_K\}}{\text{maximize}} \quad i_{EQ}(\mathbb{P}_{\{\sum_{k=1}^{K}\mathbf{z}_{\theta_k},\mathbf{d}\}})$$

Adaptive filters can be used to reduce the bias by tracking the moments in

$$\mathbf{z}_f = k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2)) - \mathbb{E}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))] + \text{std}[k_\sigma(f(\mathbf{x}_1) - f(\mathbf{x}_2))]$$
$$\mathbf{z}_d = k_\sigma(\mathbf{d}_1 - \mathbf{d}_2) - \mathbb{E}[k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)] + \text{std}[k_\sigma(\mathbf{d}_1 - \mathbf{d}_2)]$$
$$i_{EQ}(\mathbb{P}_{\{f(\mathbf{x}),\mathbf{d}\}}) = -\log_2\left[\mathbb{E}[\mathbf{z}_f]\cdot\mathbb{E}[\mathbf{z}_d]\right] + \log_2\left[\mathbb{E}[\mathbf{z}_f\cdot\mathbf{z}_d]\right]$$

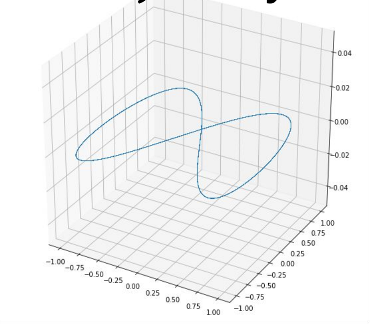
5. Results - dataset

Frequency doubler:

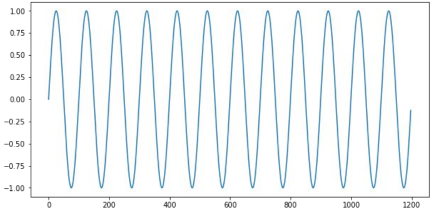
$$x_n = \sin(0.02 \cdot \pi n)$$
$$d_n = \sin(0.04 \cdot \pi n)$$

order = 3

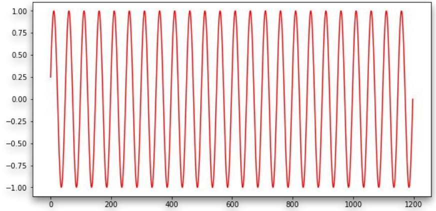
Trajectory



input



target



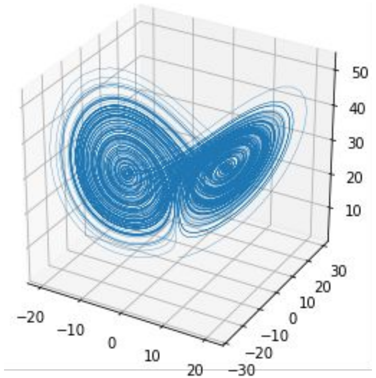
Lorenz system:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$
$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

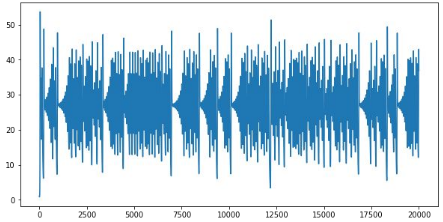
$$\{\sigma = 10, \rho = 28, \beta = 2.667\}$$
$$\{x_0 = 0, y_0 = 1, z_0 = 1.05\}$$

order = 10

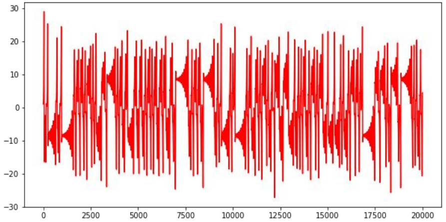
Trajector



input



target



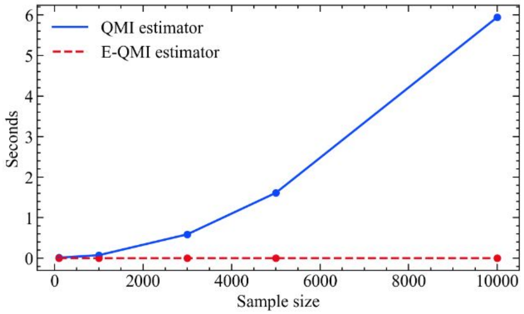
Performance compared with TDNN:

	LORENZ			FD		
	MSE	EQMI (T-I)	EQMI (T-II)	MSE ($\times 10^{-4}$)	EQMI (T-I)	EQMI (T-II)
TDNN	0.017	0.156	0.784	8.0	0.222	0.999
BWMs (T-I)	0.022	0.164	0.763	7.0	0.225	0.999
BWMs (T-II)	0.017	0.157	0.791	7.8	0.222	0.999

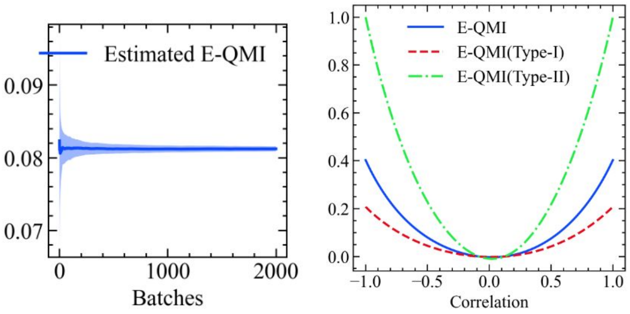
Our algorithm is either equivalent or outperforms TDNN.

BWM does not require backpropagation!

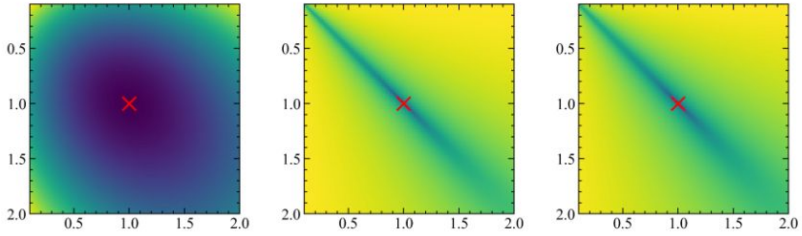
Speed of empirical embeddings:



Comparison of different normalizations:



Comparison with MSE:



(a) MSE (b) Type-I ($\sigma = 0.4$) (c) Type-II

END