Classifying Human Driving Behavior via Deep Neural Networks

Jae Hoon Kim

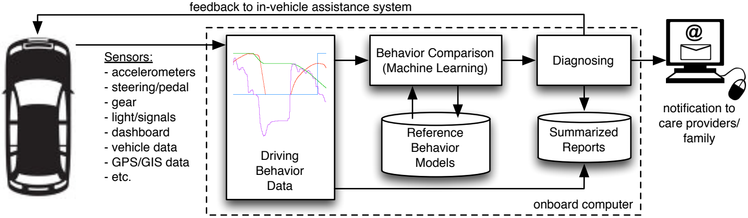
Drexel University

June 8, 2017

Table of Contents

Table of Contents

Problem Statement



- Modern car systems collect real-time data of the car status and driving behavior of the driver
- Such data is very valuable for healthcare and research
- Goal: use driving behavior data to diagnose medical conditions

Project Background

- Work in the context of the NSF SCH *Diagnostic Driving* project
- Partners:
  - Drexel University (lead)
  - Children's Hospital of Philadelphia (CHOP)
  - George Mason University
  - University of Central Florida
- CHOP and GM are in charge of data collection, and Drexel and UCF are in charge of machine learning

Project Tasks

- Learn and understand LSTM and Auto-encoders neural networks
- Classify driving simulation data from novice and expert drivers using several different neural network models
- Compare result from three different neural network models created specifically for this task

Table of Contents

Neural Networks - Introduction

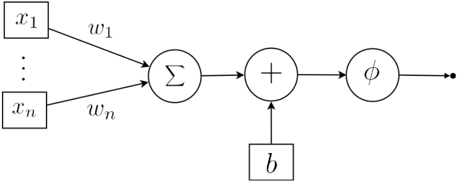


Figure: General Artificial Neuron

- An artificial neuron has input signals, weights, bias, and activation function [Ras15]

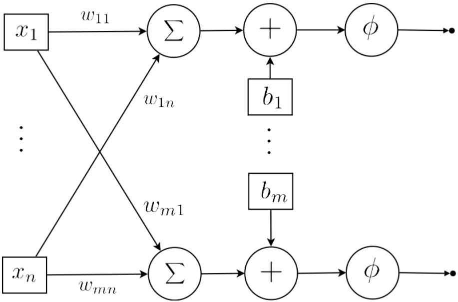


Figure: Example of a Neural Network

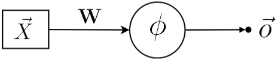


Figure: Simplified Neural Network

- Neural network is a network made up many artificial neurons

Neural Networks - Forward Propagation

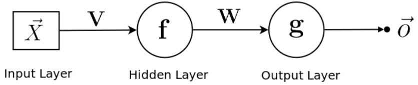


Figure: One Hidden Layer MLP

$$\vec{h} = \mathrm{f}(\vec{x}\mathrm{v} + \vec{c}) = \mathrm{f}(\vec{X}\mathrm{V})$$

$$\vec{o} = \mathrm{g}(\vec{h}\mathrm{w} + \vec{b}) = \mathrm{g}(\vec{H}\mathrm{W})$$

$$\vec{o} = \mathrm{g}(\mathrm{h}(\vec{x}\mathrm{v} + \vec{c})\mathrm{w} + \vec{b}) = \mathrm{g}(\mathrm{h}(\vec{X}\mathrm{V})\mathrm{W})$$

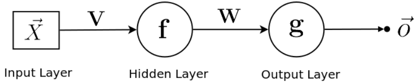


Figure: One Hidden Layer MLP

Activation Function:

$$f(z) = g(z) = sigm(z) = \frac{1}{1+e^{-z}}$$

Error Function:

$$J(\mathrm{V},\mathrm{W}) = \frac{1}{2}\sum_i (y_i - o_i)^2 = \frac{1}{2}\sum_i (y_i - g(\mathrm{h}(\vec{X}\mathrm{V})\mathrm{W_i}))^2$$

Update W_{ij} - ith neuron for jth input:

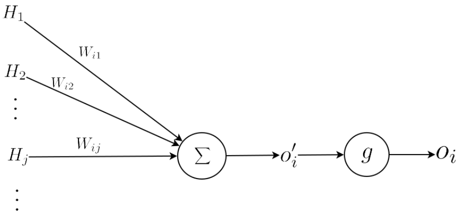


Figure: Back-propagation for W_{ij}

$$W_{ij}{}^{next} = W_{ij} - \eta \frac{\partial J}{\partial W_{ij}} = W_{ij} - \eta \frac{\partial J}{\partial o_i} \frac{\partial o_i}{\partial o_i'} \frac{\partial o_i'}{\partial W_{ij}} = W_{ij} + \eta (y_i - o_i) o_i (1 - o_i) H_j$$

Let

$$\delta_i = \frac{\partial J}{\partial o_i'} = \frac{\partial J}{\partial o_i} \frac{\partial o_i}{\partial o_i'}$$

Update V_{ij} - ith neuron for jth input:

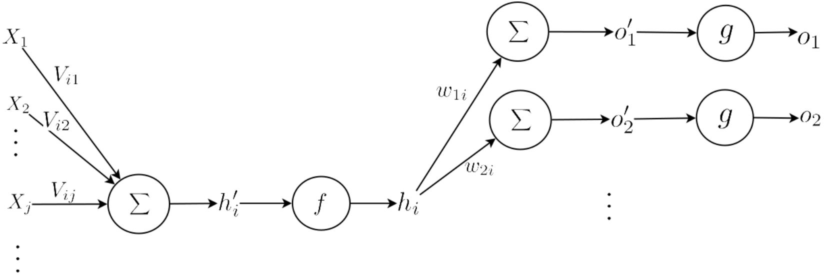


Figure: Back-propagation for V_{ij}

$$V_{ij}{}^{next} = V_{ij} - \eta \frac{\partial J}{\partial V_{ij}} = V_{ij} - \eta \frac{\partial J}{\partial h_i} \frac{\partial h_i}{\partial h_i'} \frac{\partial h_i'}{\partial V_{ij}} = V_{ij} - \eta \sum_k (\delta_k w_{ki}) h_i (1 - h_i) X_j$$

Long Short Term Memory Networks (LSTM)

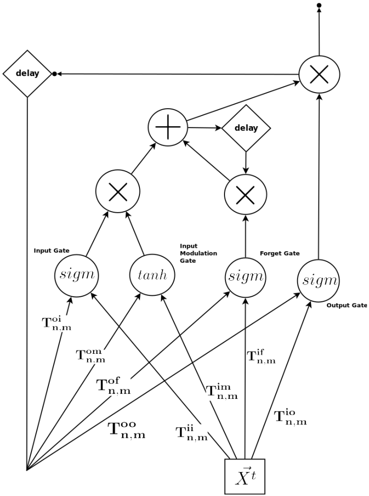


Figure: LSTM

- Recurrent Neural Network
- Four gates and Memory Cells handle long term dependencies

Long Short Term Memory Networks (LSTM)

Input gate:

$$\vec{i^t} = \mathrm{sigm}(\vec{X^t}\mathrm{T}^{\mathrm{ii}}_{\mathrm{n,m}} + \vec{H^{t-1}}\mathrm{T}^{\mathrm{oi}}_{\mathrm{m,m}})$$

Input modulation gate:

$$\vec{m^t} = \tanh(\vec{X^t}\mathrm{T}^{\mathrm{im}}_{\mathrm{n,m}} + \vec{H^{t-1}}\mathrm{T}^{\mathrm{om}}_{\mathrm{m,m}})$$

Forget gate:

$$\vec{f^t} = \mathrm{sigm}(\vec{X^t}\mathrm{T}^{\mathrm{if}}_{\mathrm{n,m}} + \vec{H^{t-1}}\mathrm{T}^{\mathrm{of}}_{\mathrm{m,m}})$$

Output gate:

$$\vec{o^t} = \mathrm{sigm}(\vec{X^t}\mathrm{T}^{\mathrm{io}}_{\mathrm{n,m}} + \vec{H^{t-1}}\mathrm{T}^{\mathrm{oo}}_{\mathrm{m,m}})$$

Memory cells:

$$\vec{c^t} = \vec{i^t} * \vec{m^t} + \vec{f^t} * \vec{c^{t-1}}$$

Result:

$$\vec{h^t} = \vec{c^t} * \vec{o^t}$$

Auto-encoder (AE) - Single layer AE

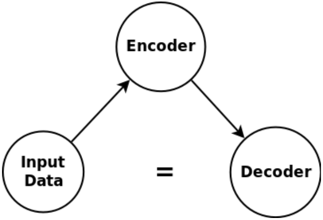


Figure: Abstract structure of Auto-encoder

Auto-encoder is a neural network to attempt to copy its inputs to its outputs

Auto-encoder (AE) - Single layer AE

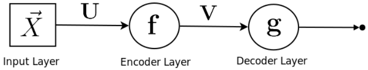


Figure: Basic Auto-encoder

$$\vec{e} = \mathrm{f}(\vec{X}\mathrm{U})$$

$$\vec{d} = \mathrm{g}(\vec{E}\mathrm{V})$$

$$E(\vec{x}, \vec{d}) = E(\vec{x}, \mathrm{g}(\vec{E}\mathrm{V})) = E(\vec{x}, \mathrm{g}(\mathrm{f}(\vec{x}\mathrm{u} + \vec{b_e})\mathrm{v} + \vec{b_d}))$$

Auto-encoder (AE) - Multiple layer AE

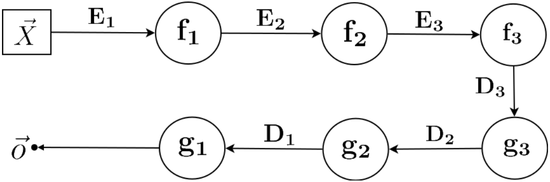


Figure: Multilayer Auto-encoder Example

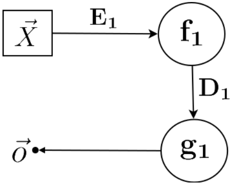


Figure: Pretraining First Step

$$E(input, data\_from\_first\_decoder)$$
$$= E(\vec{x}, \mathrm{g}_1(\mathrm{f}_1(\vec{X}\mathrm{E}_1)\mathrm{D}_1))$$

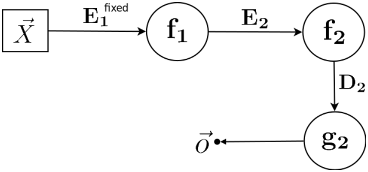


Figure: Pretraining Second Step

$$E(data\_from\_first\_encoder, data\_from\_second\_decoder)$$
$$= E(\mathrm{f}_1(\vec{X}\mathrm{E}_1^{\mathrm{fixed}}), \mathrm{g}_2(\mathrm{f}_2(\mathrm{f}_1(\vec{X}\mathrm{E}_1^{\mathrm{fixed}})\mathrm{E}_2)\mathrm{D}_2))$$

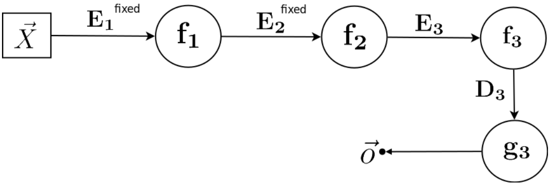


Figure: Pretraining Thrid Step

$$E(data\_from\_second\_encoder, data\_from\_third\_decoder)$$
$$= E(\mathrm{f}_2(\mathrm{f}_1(\vec{X}\mathrm{E}_1^{\mathrm{fixed}})\mathrm{E}_2^{\mathrm{fixed}}), \mathrm{g}_3(\mathrm{f}_3(\mathrm{f}_2(\mathrm{f}_1(\vec{X}\mathrm{E}_1^{\mathrm{fixed}})\mathrm{E}_2^{\mathrm{fixed}})\mathrm{E}_3)\mathrm{D}_3))$$

Auto-encoder (AE) - Multiple layer AE

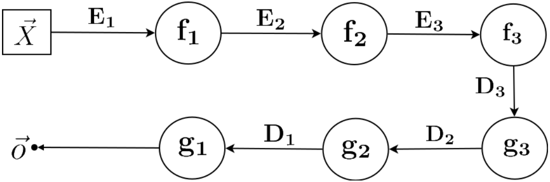


Figure: Multilayer Auto-encoder Example

$$E(input, output)$$
$$= E(\vec{x}, \mathrm{g}_1(\mathrm{g}_2(\mathrm{g}_3(\mathrm{f}_3(\mathrm{f}_2(\mathrm{f}_1(\vec{X}\mathrm{E}_1)\mathrm{E}_2)\mathrm{E}_3)\mathrm{D}_3)\mathrm{D}_2)\mathrm{D}_1))$$

Table of Contents

Figure: Driving Simulator at CHOP

Data Set

- Collected in the high-fidelity simulator of the Center for Injury Research Prevention Studies at the *Children's Hospital of Philadelphia* (CHOP)
- 16 traces $=$ 4 tracks $\times$ (2 expert drivers $+$ 2 inexpert drivers)
- 100 features collected at 60Hz
  - Car status: velocity, steer, Brake, throttle and etc.
  - Environment status: current speed limit, and etc.
  - instruction: left turn, right turn, and etc.
- Two datasets
  - *raw dataset*: 98 features without time stamps
  - *filtered dataset*: 23 features - more important features from 100 features

Data Set

Table: Length of each of the traces

Trace	*Driver*	*Track*	*Length*
Trace0	Expert0	Track0	50029
Trace1	Expert0	Track1	26375
Trace2	Expert0	Track2	29629
Trace3	Expert0	Track3	26298
Trace4	Expert1	Track0	51295
Trace5	Expert1	Track1	26674
Trace6	Expert1	Track2	29680
Trace7	Expert1	Track3	27075
Trace8	Inexpert0	Track0	49691
Trace9	Inexpert0	Track1	30058
Trace10	Inexpert0	Track2	26441
Trace11	Inexpert0	Track3	27373
Trace12	Inexpert1	Track0	47658
Trace13	Inexpert1	Track1	29380
Trace14	Inexpert1	Track2	26684
Trace15	Inexpert1	Track3	27255

- Min: 26298
  (about 7 minutes 18 seconds)
- Max: 51295
  (about 14 minutes 15 seconds)
- Average: 33224.6875
  (about 9 minutes 13 seconds)

Table of Contents

Cross Validation & Normalization

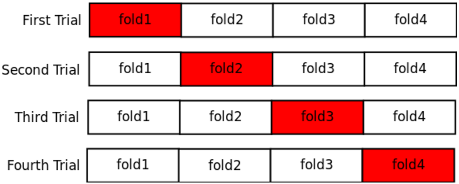


Figure: 4 Folds Cross Validation

- Cross Validation
  - To validate experiment models when data is not large enough
  - Divides a data set into K folds
  - Uses the ith fold as the test set and other folds as the training set
- Normalization
  - Normalize training set (mean 0, std 1)
  - Normalize test set by mean and standard deviation of training set

First Model

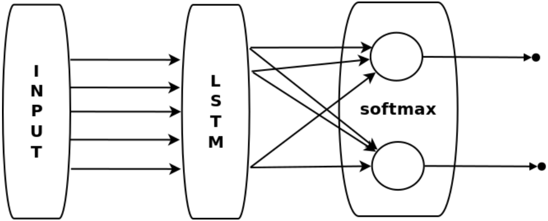


Figure: First experiment NN

- Hidden neurons in LSTM: 16, 32, 64, 128, and 256
- Without Auto-encoder on *filtered dataset*

Second Model

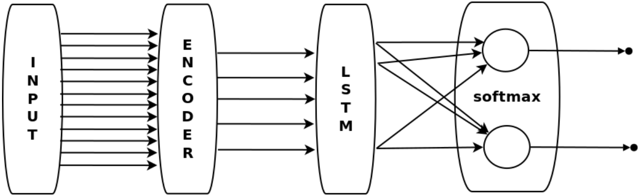


Figure: Second experiment NN

- Hidden neurons in LSTM: 16, 32, 64, 128, and 256
- With Single layer Auto-encoder on *raw dataset*:
  98 features → 25 features

Third Model

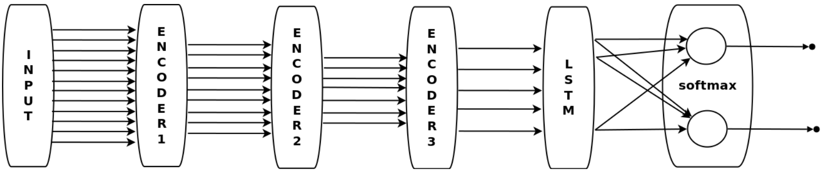


Figure: Third experiment NN

- Hidden neurons in LSTM: 16, 32, 64, 128, and 256
- With Three layer Auto-encoder on *raw dataset*:
  98 features → 75 features → 50 features → 25 features

Sampling

- Period
  - 1 over 10
  - 1 over 20
  - 1 over 50
- Sampling Methods
  - Last sample from each period
  - Mean of each period
  - Gaussian filtered value of each period

Table: Sampling and Models

Period	Method	Model
1 over 10	last	*first, second, third*
	mean	*first*
	gaussian	*first*
1 over 20	last	*first*
	mean	*first*
	gaussian	*first*
1 over 50	last	*first*
	mean	*first*
	gaussian	*first*

Table of Contents

First Model with 1 over 50

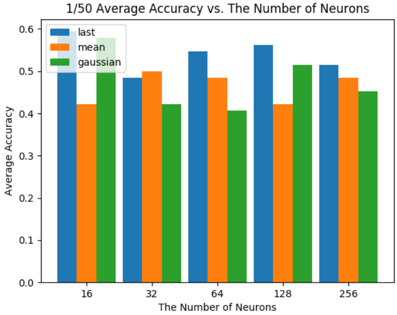


Figure: Result of first model with 1 over 50 *filtered dataset*

- Best: 0.59375 (59.375%) on *last* and 16 neurons
- Worst: 0.40625 (40.625%) on *gaussian* and 64 neurons
- Average: 0.492708 (49.2708%)

First Model with 1 over 20

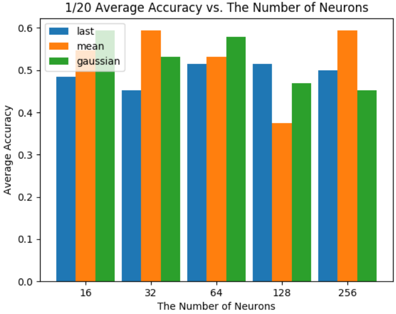


Figure: Result of first model with 1 over 20 *filtered dataset*

- Best: 0.59375 (59.375%) on *mean* and 32 and 256 neurons
- Worst: 0.375 (37.5%) on *mean* and 128 neurons
- Average: 0.515625 (51.5625%)

First Model with 1 over 10

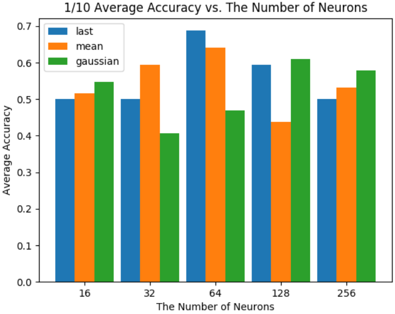


Figure: Result of first model with 1 over 10 *filtered dataset*

- Best: 0.6875 (68.75%) on *last* and 64 neurons
- Worst: 0.40625 (40.625%) on *gaussian* and 32 neurons
- Average: 0.540625 (54.00625%)

Comparison of three models with 1 over 10, *last* method

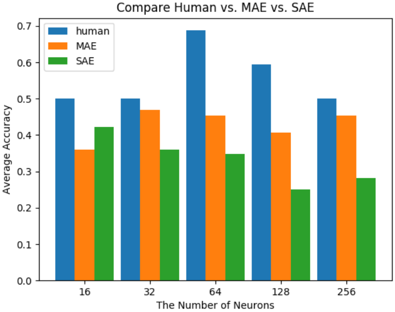
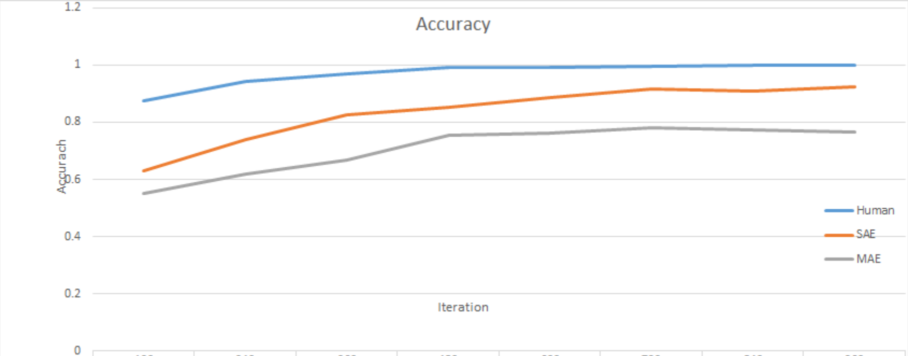


Figure: Result of three models with 1 over 10, *last* method

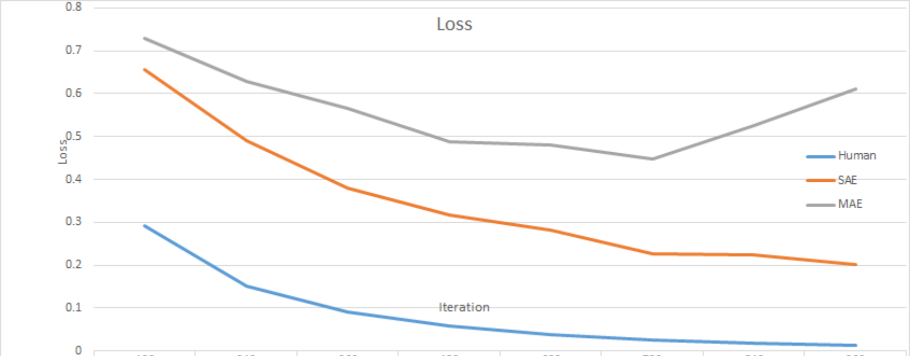
Models with Auto-encoder show worse performance

Comparison of three models with 1 over 10, *last* method



	120	240	360	480	600	720	840	960
Human	0.875000125	0.941666875	0.970833375	0.99062525	0.99375025	0.994791875	0.99791675	0.998958375
SAE	0.63125	0.738541625	0.827083875	0.854166875	0.885417	0.91562525	0.90833375	0.925000375
MAE	0.55	0.61874975	0.66666625	0.756250125	0.76354175	0.780208375	0.7729165	0.765625125

Figure: Three Models Training Accuracy



	120	240	360	480	600	720	840	960
Human	0.292777775	0.15116075	0.089749625	0.057490588	0.037923075	0.02640095	0.018739875	0.01357385
SAE	0.65691935	0.490994663	0.3801422	0.3182443	0.283100488	0.225754163	0.224616588	0.20157165
MAE	0.72904015	0.628999963	0.565557138	0.4877104	0.480205938	0.448890388	0.525507238	0.612070738

Figure: Three Models Training Loss

Problems with second and third models

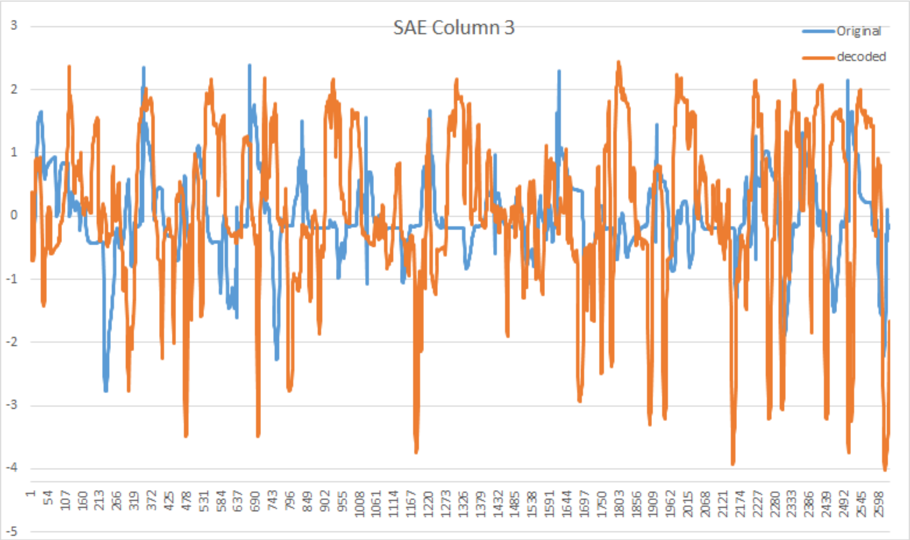


Figure: SAE, AE Error for Column 3

Problems with second and third models

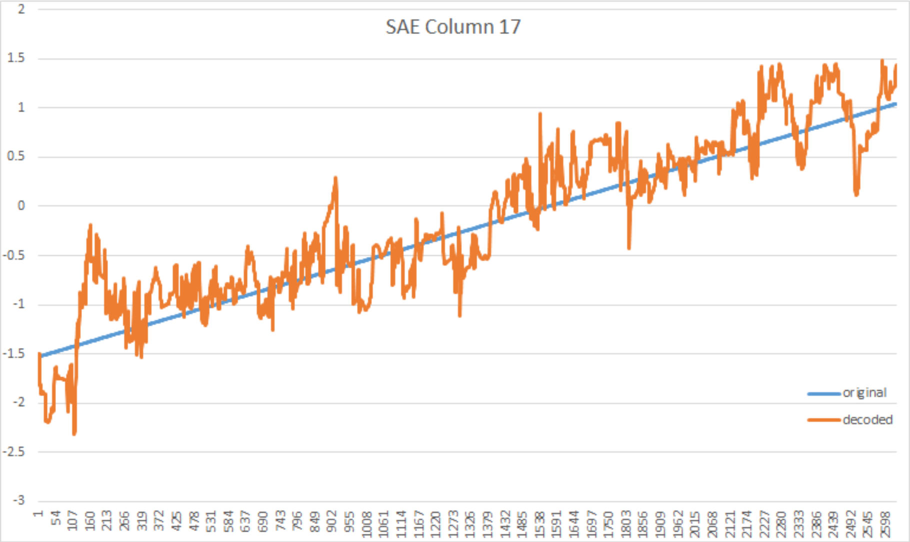


Figure: SAE, AE Error for Column 17

Problems with second and third models

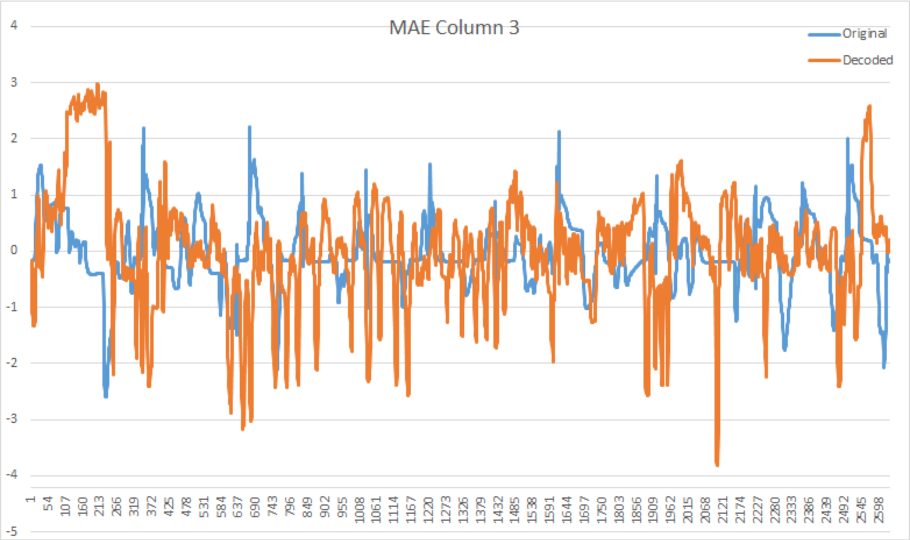


Figure: MAE, AE Error for Column 3

Problems with second and third models

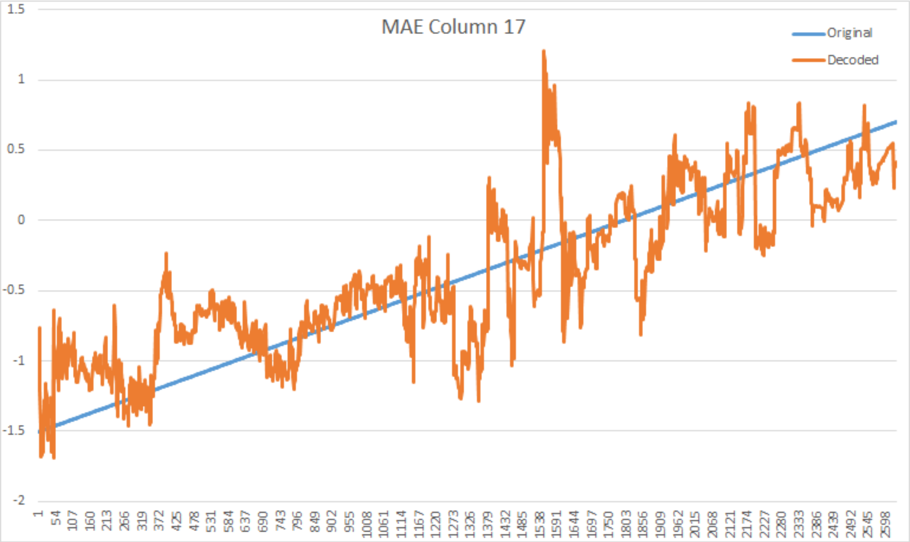


Figure: MAE, AE Error for Column 17

Table of Contents

Conclusion - Data Size

- The number of samples in trace:
  From the result of experiments with different re-sampled periods, more densely re-sampled data gives better performance
- The number of traces:
  12 traces are not enough to train the proposed neural networks

Conclusion - Limitation of Auto-encoders

- No optimize transfer matrices:
  When Auto-encoder tries to reduce dimensions, if encoder and decoder transfer matrices do not exist, encoded data has noise. It can make it hard to train subsequent neural networks.
- Always try to keep all information:
  Auto-encoder does not ignore or exclude unimportant features (it's unsupervised)

Conclusion - Problem in training MAE

- Training time:
  Processing of pretraining takes a long time
- Learn error from previous layer during pretraining:
  Fix previous layer while the following auto-encoder layer is trained

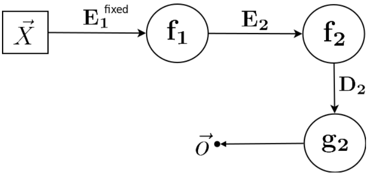


Figure: Pretraining Second Step

Table of Contents

Future Work

- Experiment with more traces
- Research auto-encoder that accept some form of supervision and reduce dimensions with the ability to ignore unimportant features and keep information from the important features (and that can be used for sequential data with LSTMs)
- Try different ways to train MAE

Future Work

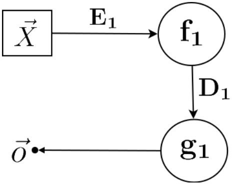


Figure: First Step of New Way to train MAE

$$E(input, output)$$
$$= E(\vec{x}, \mathrm{g}_1(\mathrm{f}_1(\vec{X}\mathrm{E}_1)\mathrm{D}_1))$$

Future Work

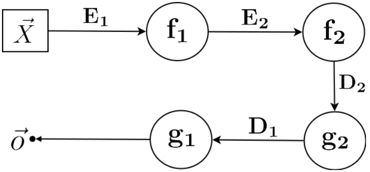


Figure: Second Step of New Way to train MAE

$$
\begin{aligned}
& E(input, output) \\
= & E(\vec{x}, \mathrm{g}_1(\mathrm{g}_2(\mathrm{f}_2(\mathrm{f}_1(\vec{X}\mathrm{E}_1)\mathrm{E}_2)\mathrm{D}_2)\mathrm{D}_1))
\end{aligned}
$$

Future Work

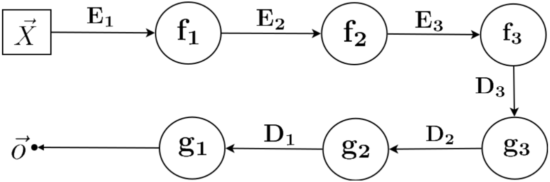


Figure: Thrid Step of New Way to train MAE

$$E(input, output)$$
$$= E(\vec{x}, \mathrm{g}_1(\mathrm{g}_2(\mathrm{g}_3(\mathrm{f}_3(\mathrm{f}_2(\mathrm{f}_1(\vec{X}\mathrm{E}_1)\mathrm{E}_2)\mathrm{E}_3)\mathrm{D}_3)\mathrm{D}_2)\mathrm{D}_1))$$

Thank You

References I

Sebastian Raschka, *Python machine learning*, Packt Publishing Ltd, 2015.