Sep. 23, 2022
The 21th Behavior Modeling in Transportation Networks

Fundamentals of neural network modeling and its applications to travel behavior analysis

Hiroshima University
Makoto Chikaraishi

Contents

1. Fundamentals of neural network modeling

2. Applications to travel behavior analysis

Main reference: Goodfellow, I., Bengio, Y., Courville, A. (2016) Deep Learning, MIT Press.

Recommend to read this textbook to understand the details!

1. FUNDAMENTALS OF NEURAL NETWORK MODELING

Deep feedforward networks (FFN)

- Also called as multilayer perceptrons
- Goal of using (deep) FFN
  - Approximate some function f that maps an input x to output y:

$$y = f^*(\boldsymbol{x})$$

  - Specifically, estimate parameters $\boldsymbol{\theta}$ that result in the best function approximation f

$$y = f(\boldsymbol{x}; \boldsymbol{\theta})$$

- It's called feedforward, since no feedback connections:

$$f(\boldsymbol{x}; \theta) = f^{(3)} f^{(2)}(f^{(1)}(\boldsymbol{x}; \boldsymbol{\theta})))$$

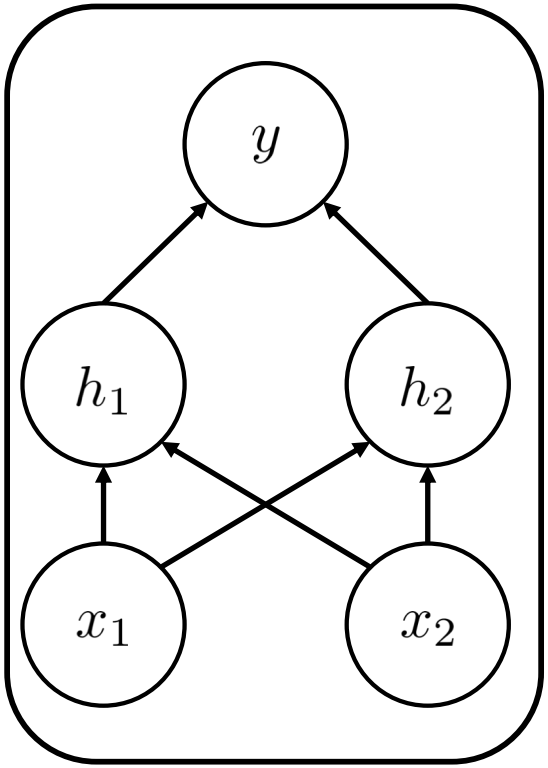
↑ Output layer ↑ Second layer ↑ First layer

Why (Deep) NN?

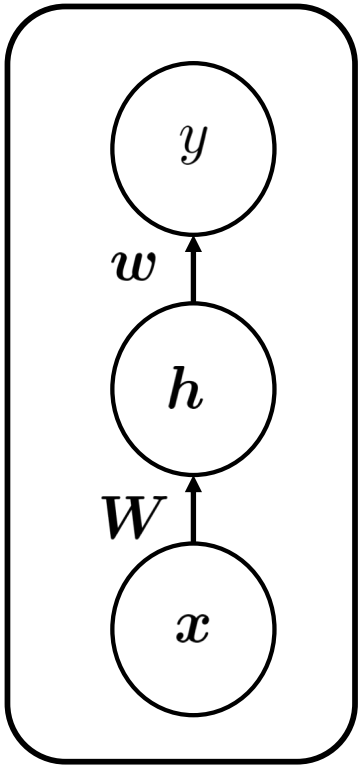
- Limitations of linear regression
  - Cannot consider 1) non-linearity of input variables & 2) interactions among input variables

- Strategies: Non-linear transformation of $\boldsymbol{x}$: $\phi(\boldsymbol{x})$
  1. Use very generic $\phi(\boldsymbol{x})$ (e.g., Kernel)
  2. Manually tune $\phi(\boldsymbol{x})$ (using expert knowledge)
  3. Use neural network to tune $\phi(\boldsymbol{x})$

An example of FFN



Different notation



$\boldsymbol{W}$: mapping from $\boldsymbol{x}$ to $\boldsymbol{h}$ (parameters)

$\boldsymbol{w}$: mapping from $\boldsymbol{h}$ to y (parameters)

An example:

Rectified linear unit (ReLU)

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b$$

Affine transformation

Designing and training a NN

1. Define a cost function

e.g., using maximum likelihood (i.e., the cross-entropy between the training data and the model distribution)

$$J(\boldsymbol{\theta}) = -E_{\boldsymbol{x},\boldsymbol{y}\sim\hat{p}_{data}} \log p_{model}(\boldsymbol{y}|\boldsymbol{x})$$

Note: there are several other cost functions, including mean squared error, mean absolute error, etc.

Designing and training a NN

2. Output units

- Linear units for Gaussian output distributions
  - Only use an affine transformation

$$\hat{y} = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{h} + b$$

- Sigmoid units for Bernoulli output distributions
  - Affine transportation + logit transformation

$$\hat{y} = \sigma(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{h} + b) \quad \text{where} \quad \sigma(x) = \frac{1}{1+\exp(-x)}$$

- Softmax units for Multinoulli output distibutions
  - Affine transformation + MNL transformation

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad \text{where} \quad \boldsymbol{z} = \boldsymbol{W}^{\mathsf{T}}\boldsymbol{h} + \boldsymbol{b}$$

Designing and training a NN

3. Hidden units

- Activation functions are used on top of an affine transformation:

$$\boldsymbol{h} = g(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{b})$$

- A widely used activation function:
  - ReLU: Rectified Linear Unit

$$g(z) = \max\{0, z\}$$

- Generalizations of ReLU
  - Using nonzero slope α_i when z<0

$$h_i = g(\boldsymbol{z}, \boldsymbol{\alpha})_i = \max\{0, z_i\} + \alpha_i \min\{0, z_i\}$$

1. Absolute value rectification: $\alpha_i = -1$
2. Leaky ReLU: Give a small fixed value to α_i such as 0.01
3. Parametric ReLU: treat α_i as a learnable parameter

- There are many other activation functions

Architecture design

- Architecture refers to the overall structure of the network. Most NNs are organized into groups of units called layers.

$$\boldsymbol{h}^{(1)} = g^{(1)}(\boldsymbol{W}^{(1)\mathsf{T}}\boldsymbol{x} + \boldsymbol{b}^{(1)})$$

$$\boldsymbol{h}^{(2)} = g^{(2)}(\boldsymbol{W}^{(2)\mathsf{T}}\boldsymbol{x} + \boldsymbol{b}^{(2)})$$

$$\vdots$$

- Main architectural considerations
  - The depth of the network and the width of each layer
  - Whether to have skip connections or not
  - How to connect a pair of layers to each other (**W**)

  EX) Convolutional NN, Recurrent NN
- The ideal network architecture for a task must be found via experimentation guided by monitoring the validation set error.

Universal approximation theorem

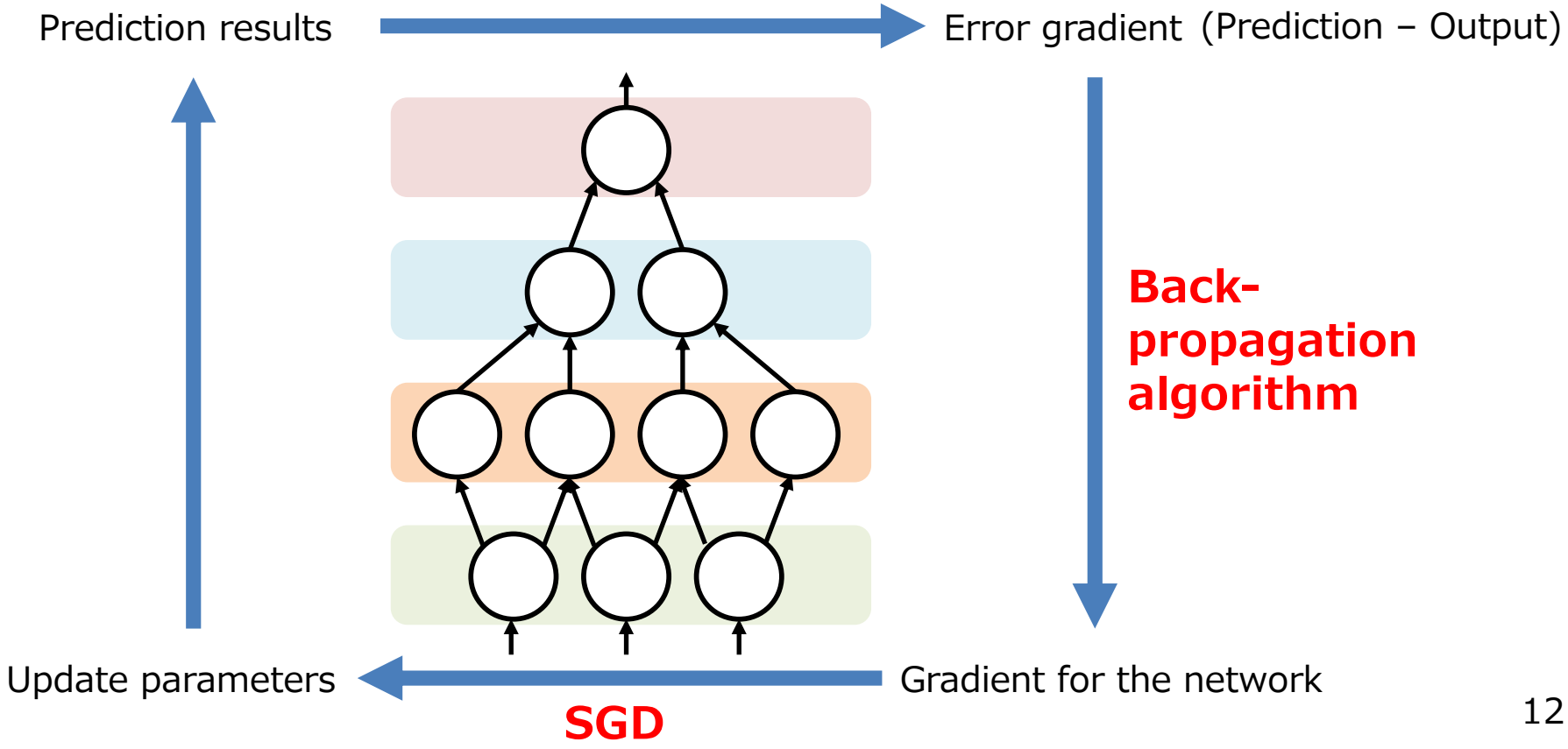
- Universal approximation theorem by Hornik et al. (1989), Cybenko (1989)
  - This theorem says that neural networks can approximate any function.

- This theorem also said that “shallow” network structure can approximate any function, while it is also known that more efficient learning can be achieved with “deep” network structure.

- Another important issue is the interpretability of the fully connected NN.

Learning algorithm

- **Back-propagation algorithm** (Rumelhart et al., 1986)
  - Computing the gradient
- **Stochastic gradient descent (SGD)**
  - Perform learning using the gradient



Stochastic gradient descent (SGD)

- Conventional gradient descent

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \theta)$$

$$\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \theta)$$

- Stochastic gradient descent
  - Use mini-batch m' (small subset of training data) to compute the gradient

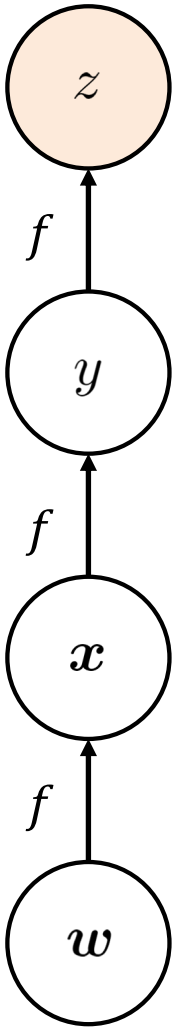
$$g = \frac{1}{m'} \sum_{i=1}^{m'} \nabla_\theta L(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \theta)$$

$$\theta \leftarrow \theta - \epsilon g$$

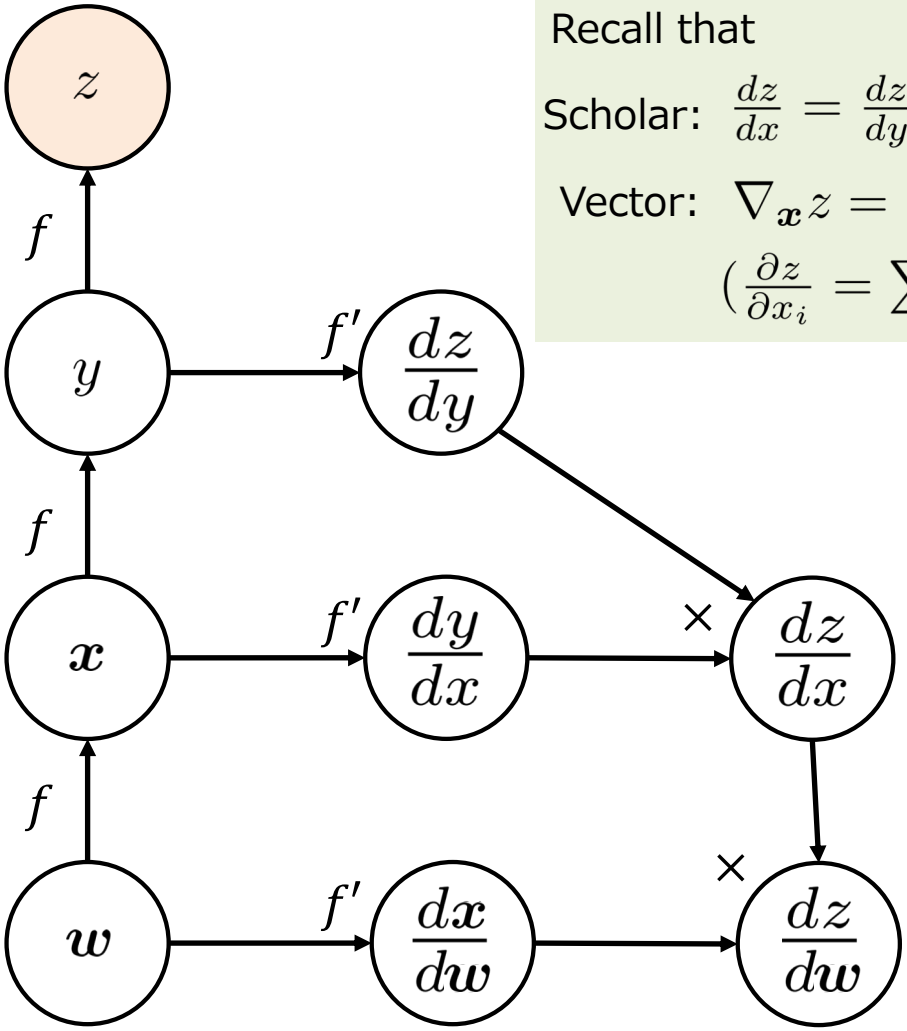
Learning rate

Back-propagation algorithm

Computational graph of NN



Add additional nodes of gradients



Recall that

Scholar: $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Vector: $\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\mathsf{T}} \nabla_{\boldsymbol{y}} z$

$\left(\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_i}\frac{\partial y_i}{\partial x_i}\right)$

Regularization

- **What is regularization**
  - "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error"
- **Some methods**
  1. Put extra constraints on a ML model
  2. Add extra terms in the objective function
  3. Ensemble methods, combining multiple hypotheses that explain the training data.
- **Fundamental properties**
  - Generally increasing bias for reducing variance.

Parameter norm penalities

- Regularized objective function

$$\tilde{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\theta)$$ The norm penalty term

- L^2 parameter regularization (ridge regression)

$$\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = J(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha\mathbf{w}^\top\mathbf{w}$$

- L^1 parameter regularization (LASSO)

$$\tilde{J}(\mathbf{w}; \mathbf{X}, \mathbf{y}) = J(\mathbf{w}; \mathbf{X}, \mathbf{y}) + \alpha \sum_i |w_i|$$

- Elastic net

2. APPLICATIONS TO TRAVEL BEHAVIOR ANALYSIS

Applications of deep learning method in transport studies

	Accidents	Congestion	DB	M–AC	TD	TTP	Dist	Flow	Speed	Occ	Total
CNN	36	23	38	4	20	10	0	43	101	3	278
CNN-GRU	0	0	2	0	0	0	0	9	18	0	29
CNN-LSTM	3	2	2	0	9	3	0	11	18	0	48
CNN-RNN	0	0	0	0	0	0	0	0	10	0	10
LSTM- GRU	0	0	0	0	0	0	0	0	1	0	1
DBN	2	2	5	0	0	44	0	50	10	1	114
DNN	16	1	5	2	10	2	2	38	37	0	113
GRU	0	8	0	0	3	2	0	2	33	0	48
LSTM	10	16	12	4	69	11	0	77	90	0	289
RNN	1	9	12	1	1	4	0	12	20	0	60
SAE	4	1	0	0	14	20	0	151	32	0	222
TM	59	27	46	19	90	35	1	324	218	1	820
SNN	7	14	7	1	42	6	0	142	62	1	282
Total	138	103	129	31	258	137	3	859	650	6	2314

Abbreviations for area of application: DB, driver behaviour; M–AC, mode and activity choice; TD, travel demand; TTP, travel time prediction; Dist., travel distance; Occ, occupancy

Varghese, V. Chikaraishi, M., Urata, J. (2020) Deep Learning in Transport Studies: A Meta-Analysis on the Prediction Accuracy, Journal of Big Data Analytics in Transportation, Vol. 2, 199–220.

Tension between theory-driven methods (classical choice models) and data-driven methods (machine learning)

- **Winner of the 2018 Eric Pas Best Dissertation Award**
  - Timothy Brathwaite
    - The Holy Trinity: Blending Statistics, Machine Learning and Discrete Choice with Applications to Strategic Bicycle Planning

- **ICMC2019 keynote**
  - Joan Walker
    - Choice modelling in an age of machine learning

- **Honorable Mention of the 2019 Eric Pas Best Dissertation Award**
  - Shenhao Wang
    - Deep neural networks for choice analysis

And many others...

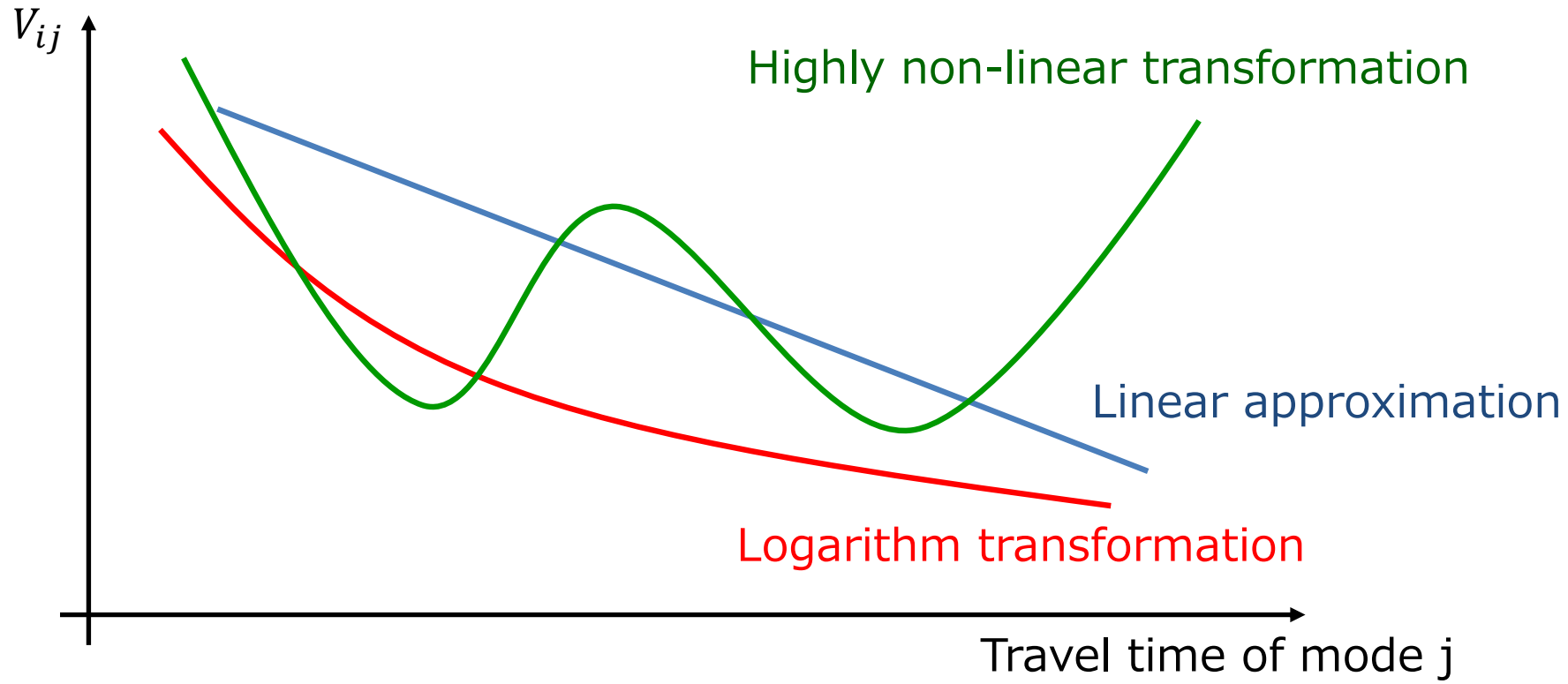
Problem setting

- Standard logit model: $P_{ij} = \frac{\exp(V_{ij})}{\sum_{j'=1}^{J} \exp(V_{ij'})}$
- The conventional form of V_{ij}:
  - Linear approximation (rooted to the Taylor's theorem)
  - Also known as a linear-in-parameter model
- Problem at hand:
  - Is there any better way to determine the functional form?
    - Obviously, taking into account the non-linearity of V_{ij} would improve the goodness-of-fit.
    - What is the cost of doing that?

Problem setting

- Can we understand the non-linear transformation of V_{ij} logically?

Example: contribution of travel time to mode/route choice model



Non-linearity through neural network (NN): It's about how to construct network architecture

A classical MNL

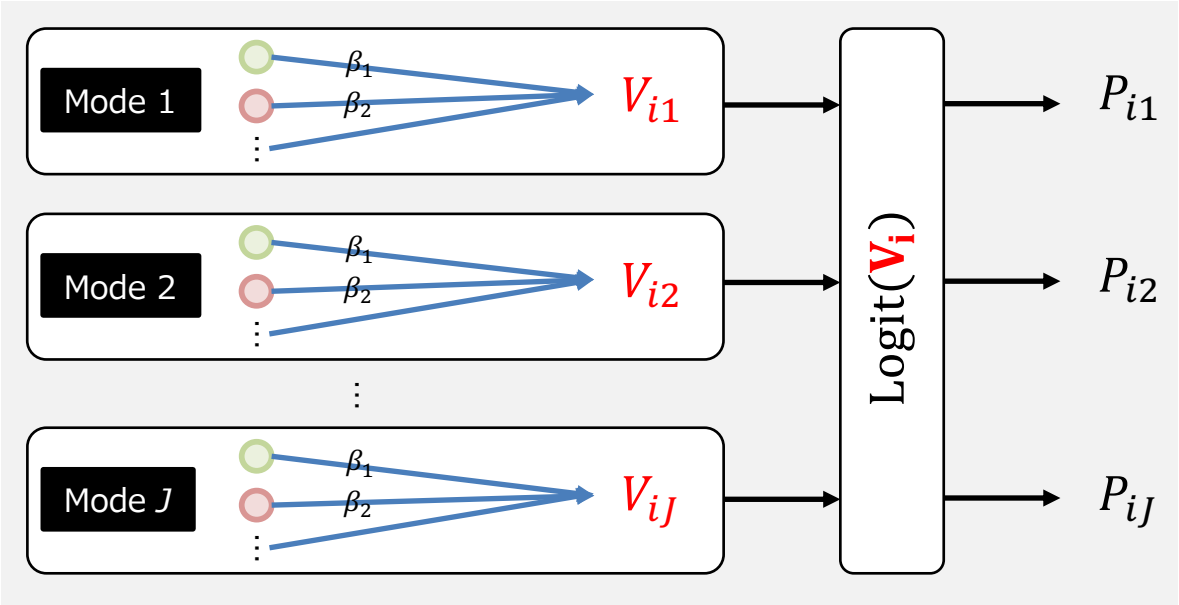
$$P_{ij} = \text{Logit}(\mathbf{V_i}) = \frac{\exp(V_{ij})}{\Sigma_{j'=1}^{J} \exp(V_{ij'})}$$

$$V_{ij} = \beta_1 x_{ij1} + \beta_2 x_{ij2} + \cdots$$

Travel time of mode j

Travel cost of mode j

Network architecture



(Deep) NN

(Fully connected)

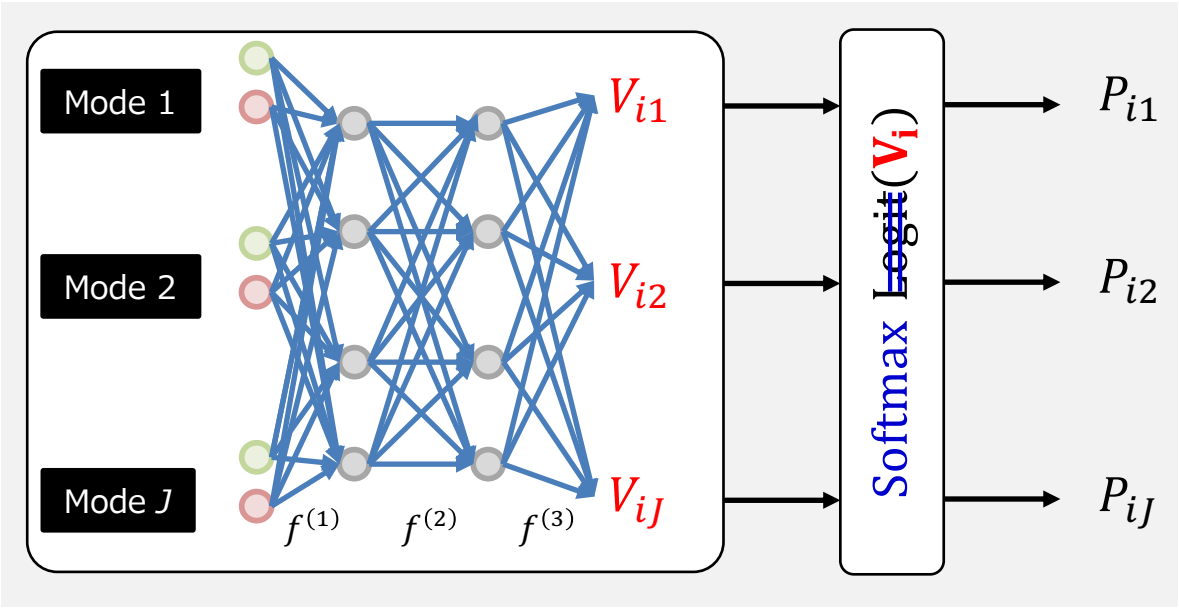
$$P_{ij} = \frac{\exp(V_{ij})}{\Sigma_{j'=1}^{J} \exp(V_{ij'})}$$

$$V_{ij} = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\mathbf{x_i})\right)\right)$$

An example of f: Rectified linear unit (ReLU)

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^{\mathsf{T}} \max\{0, \boldsymbol{W}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{c}\} + b$$

Network architecture



Fully connected DNN often does not work well (and produce less explainable results)

Seeking a better network structure in between.

A classical MNL

$$P_{ij} = \mathrm{Logit}(\mathbf{V_i}) = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij} = \beta_1 x_{ij1} + \beta_2 x_{ij2} + \cdots$$

Travel time of mode j

Travel cost of mode j

(Deep) NN

(Fully connected)

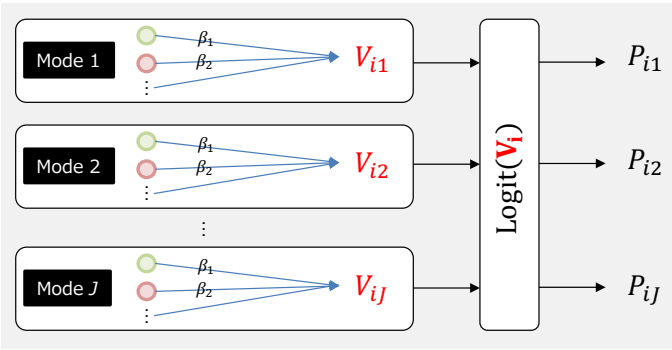
$$P_{ij} = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij} = f^{(3)}\left(f^{(2)}\left(f^{(1)}(\mathbf{x_i})\right)\right)$$

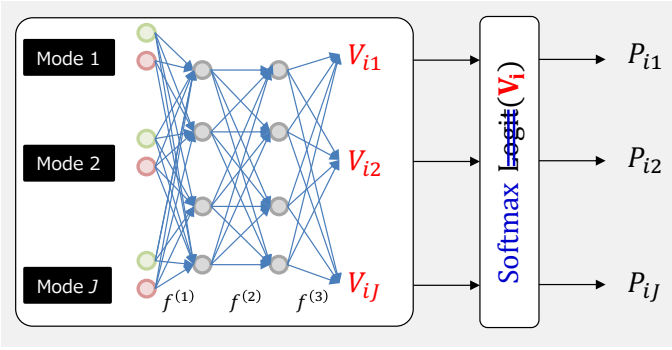
An example of f: Rectified linear unit (ReLU)

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^{\top} \max\{0, \boldsymbol{W}^{\top}\boldsymbol{x} + \boldsymbol{c}\} + b$$

Network architecture



Network architecture



- ✓ *Sifringer, B., Lurkin, V.and Alahi, A.: Enhancing discrete choice models with representation learning. Transportation Research Part B 140, 236-261, 2020.*
- ✓ *Wang, S.: Deep neural networks for choice analysis, PhD Dissertation at MIT,2020 https://dspace.mit.edu/handle/1721.1/129894*
- ✓ *Han Y, Pereira FC, Ben-Akiva M, Zegras C. A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. Transportation Research Part B: Methodological. 2022;163:166-86.*

SOME EFFORTS TO IMPROVE THE INTERPRETABILITY

Interpretability (Lipton, 2018)

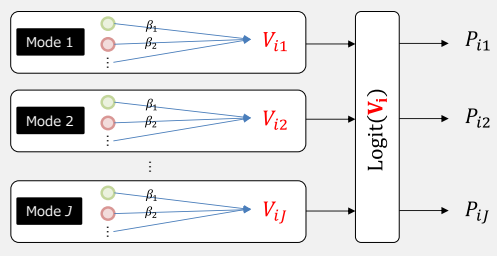
- Motivations for asking interpretability
  - trust, causality, transferability, informativeness, and fair and ethical decision making

- Two broader categories of interpretability
  - **Transparency**
    - i.e., how does the model work.
    - The opposite of “black-boxness”
  - **Post hoc explanations**
    - i.e., what else can the model tell me.
    - Extracting information from learned models.

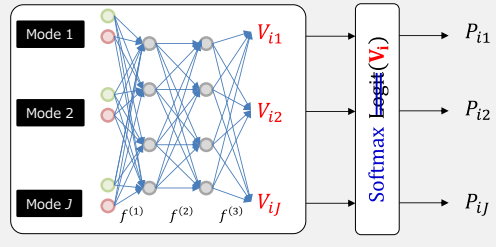
Efforts to keep both interpretability and accuracy

- **Wang (2020)**
  - From fully connected deep neural network (F-DNN) to DNN with alternative-specific utility functions (ASU-DNN)

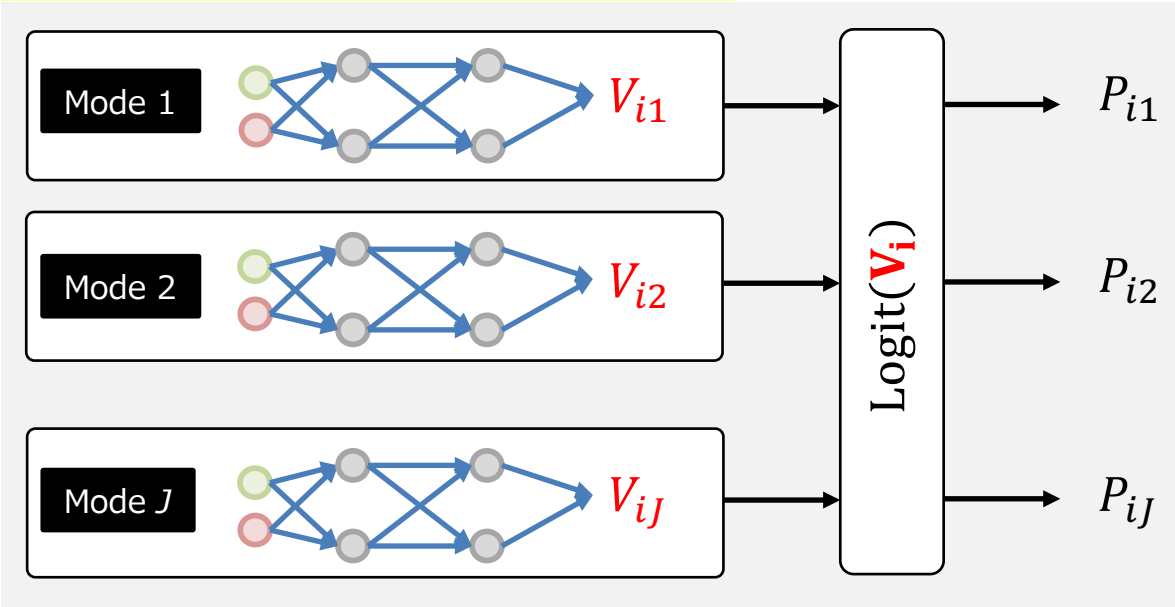
A classical MNL



(Deep) NN



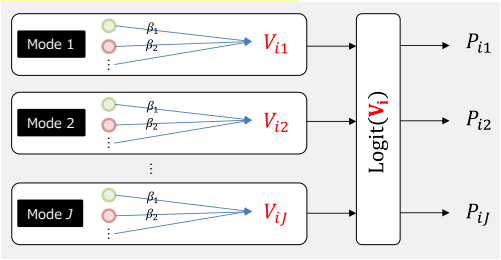
Proposed network architecture



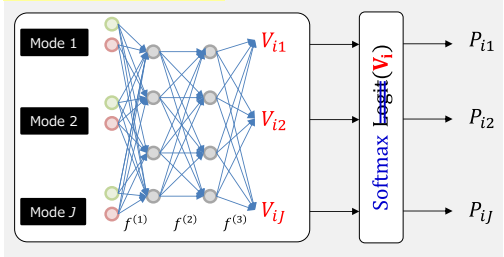
Efforts to keep both interpretability and accuracy

- **Sifringer et al. (2020)**
  - Traditional linear-in-parameters are assumed for important policy variables, while DNN is used for the rest of variables (TB-ResNets proposed by Wang (2020) also follows a similar idea, but use a different method to implement it)

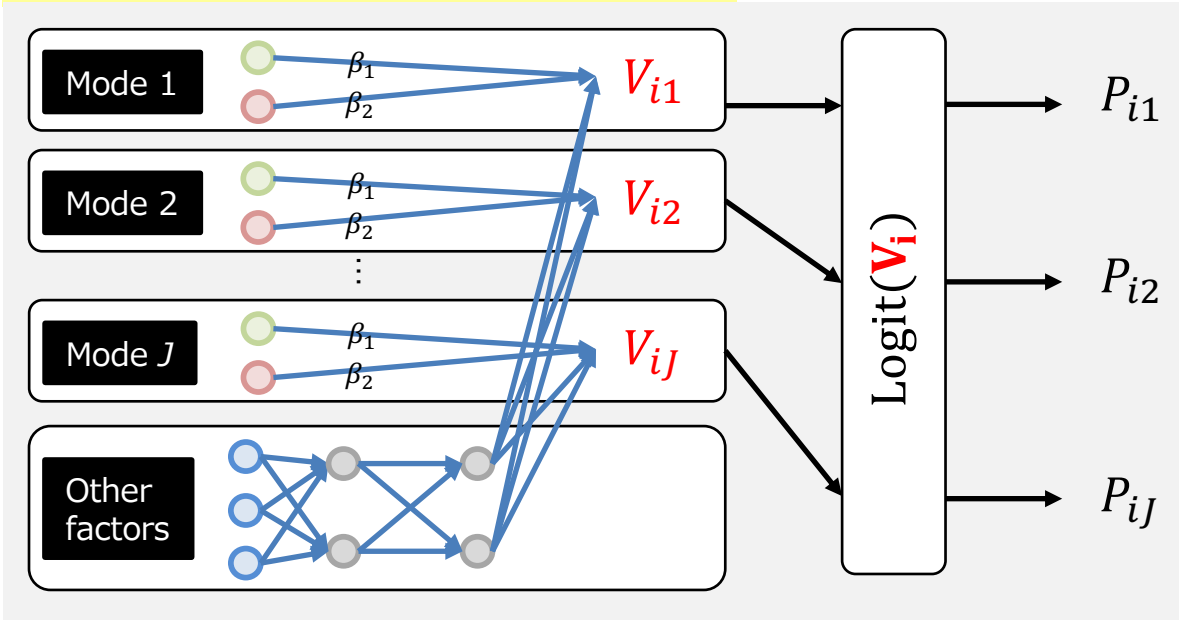
A classical MNL



(Deep) NN



Proposed network architecture



Model of Sifringer et al. (2020)

A classical MNL

$$P_{ij} = \text{Logit}(\mathbf{V_i}) = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij}^{MNL} = \beta_1 x_{ij1}^{MNL} + \beta_2 x_{ij2}^{MNL} + \cdots$$

Sifringer et al. (2020)

$$P_{ij} = \text{Logit}(\mathbf{V_i}) = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij} = V_{ij}^{MNL} + V_{ij}^{NN}$$

Note: V_{ij}^{NN} should not include variables used in V_{ij}^{MNL}

(Deep) NN

(Fully connected)

$$P_{ij} = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij}^{NN} = f^{(3)}\left(f^{(2)}\left(f^{(1)}\left(\mathbf{x_i^{NN}}\right)\right)\right)$$

An example of f: Rectified linear unit (ReLU)

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^{\mathsf{T}} \max\{0, \boldsymbol{W}^{\mathsf{T}}\boldsymbol{x} + \boldsymbol{c}\} + b$$

Han et al. (2022): Further extension of Sifringer et al. (2020)

Sifringer et al. (2020)

$$P_{ij} = \text{Logit}(\mathbf{V_i}) = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij} = V_{ij}^{MNL} + V_{ij}^{NN}$$

Han et al. (2022)

$$P_{ij} = \text{Logit}(\mathbf{V_i}) = \frac{\exp(V_{ij})}{\Sigma_{j\prime=1}^{J} \exp(V_{ij\prime})}$$

$$V_{ij} = \beta_i^{TN} x_{ij}^{TN} + \beta^{MNL} x_{ij}^{MNL}$$

$$\beta_i^{TN} = TasteNet(z_i; W)$$

Individual characteristics

Note: Han et al. (2022) and Sifringer et al. (2020) are essentially the same when x_{ij}^{TN} only contain the constant.

Taste heterogeneities are considered in Han et al. (2022), which represents through individual characteristics z_i

Relationship between Han et al. (2022) and latent class model

Mixed logit

$$P_{ij} = \int \underbrace{L_{ij}(\beta)}_{\text{Choice probability conditional on } \beta} \underbrace{f(\beta)}_{\text{Mixing distribution}} d\beta \qquad \text{where} \qquad L_{ij}(\beta) = \frac{\exp(\beta x_{ij})}{\Sigma_{j\prime=1}^{J} \exp(\beta x_{ij\prime})}$$

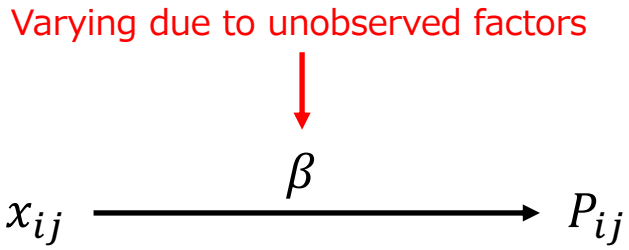
Latent class

(i.e., the discrete mixing distribution is assumed in the mixed logit)

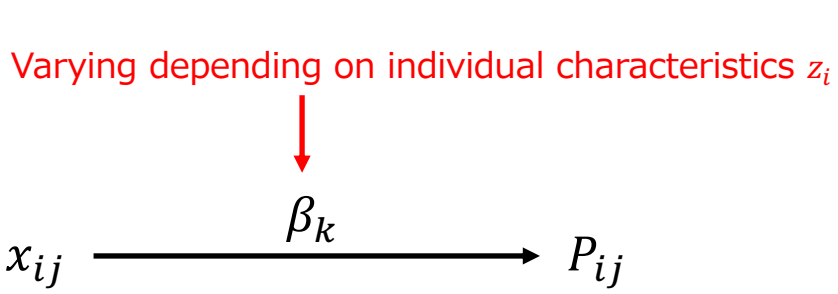
$$P_{ij} = \sum_k L_{ij}(\beta_k) \underbrace{P_{ik}(z_i)}_{\text{Probability of belonging to class } k} \qquad \text{where} \qquad L_{ij}(\beta_k) = \frac{\exp(\beta_k x_{ij})}{\Sigma_{j\prime=1}^{J} \exp(\beta_k x_{ij})}$$

From a perspective of moderation effects

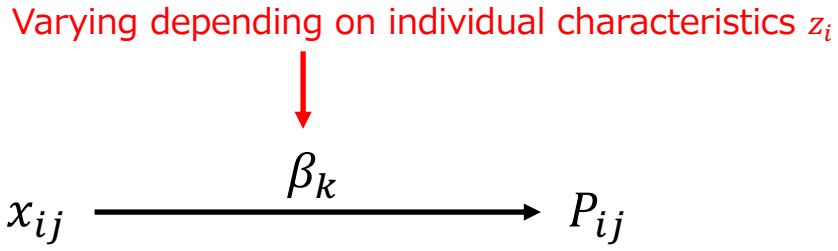
Mixed logit



Latent class



Han et al. (2022)



Latent class model is comparable to feed-forward neural network under a certain condition (Vermunt and Magidson. 2003)

From a perspective of moderation effects

- The drawback of the straightforward application of neural network is in its interpretability.
- From the latent class model perspective, this issue essentially comes from the fact that all input variables have double roles, i.e., (1) variables directly affecting the choice decisions, and (2) variables affecting the class belonging probability which plays a moderator role.

Han Y, Pereira FC, Ben-Akiva M, Zegras C. A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. Transportation Research Part B: Methodological. 2022;163:166-86.

REVISITING THE ROLE OF REGULARIZATION

Interesting finding in Han et al.

Generate synthetic datasets using “true” utility function shown on the right

When $time_{in}$ and $wait_{in}$ are correlated ($\rho = 0.6$), The performance of TasteNet-MNL could outperform that of TRUE model !!

True utility function

$$V_{in} = ASC_i - cost_{in} +$$
$$(-0.1 - 0.5inc_n - 0.1full_n + 0.05flex_n$$
$$-0.2inc_n * full_n + 0.05inc_n * flex_n + 0.1full_n * flex_n) * time_{in} +$$
$$(-0.2 - 0.8inc_n - 0.3full_n + 0.1flex_n$$
$$-0.3inc_n * full_n + 0.08inc_n * flex_n + 0.3full_n * flex_n) * wait_{in}$$

Table 4
Parameter estimates by MNLs and TasteNet-MNL (Correlated Data).

Coef	MNL-I	MNL-II	MNL-TRUE	TasteNet-MNL best run	TasteNet-MNL 100 runs: mean (std)	Truth
ASC1	−0.0335	−0.0385	−0.0543	−0.0561	−0.0627	−0.10
	(0.0282)	(0.0265)	(0.0258)		(0.0528)	
b_time	−0.0871	−0.0310	−0.0988	−0.1090	−0.0963	−0.10
	(0.0028)	(0.0033)	(0.0063)		(0.0089)	
b_time_flex	0.1211	0.0016	0.0500	0.0723	0.0592	0.05
	(0.0018)	(0.0043)	(0.0047)		(0.0088)	
b_time_full	−0.1103	−0.1113	−0.1003	−0.1030	−0.1271	−0.10
	(0.0030)	(0.0030)	(0.0080)		(0.0174)	
b_time_full_flex			0.1012	0.0861	0.0872	0.10
			(0.0054)		(0.0142)	
b_time_inc	−0.6562	−0.8004	−0.4979	−0.4836	−0.5439	−0.50
	(0.0093)	(0.0103)	(0.0241)		(0.0363)	
b_time_inc_flex		0.3087	0.0497	0.0248	0.0562	0.05
		(0.0104)	(0.0161)		(0.0283)	
b_time_inc_full			−0.2077	−0.1866	−0.1279	−0.20
			(0.0243)		(0.0410)	
b_wait	−0.2256	−0.0663	−0.1790	−0.2196	−0.2079	−0.20
	(0.0106)	(0.0125)	(0.0242)		(0.0170)	
b_wait_flex	0.2805	−0.0587	0.0951	0.0834	0.1004	0.10
	(0.0068)	(0.0161)	(0.0178)		(0.0173)	
b_wait_full	−0.1913	−0.2068	−0.3256	−0.3054	−0.3243	−0.30
	(0.0111)	(0.0110)	(0.0304)		(0.0283)	
b_wait_full_flex			0.3196	0.2856	0.2735	0.30
			(0.0201)		(0.0230)	
b_wait_inc	−1.1014	−1.4841	−0.9142	−0.8038	−0.8586	−0.80
	(0.0340)	(0.0385)	(0.0922)		(0.0680)	
b_wait_inc_flex		0.8868	0.0576	0.0886	0.0974	0.08
		(0.0388)	(0.0612)		(0.0544)	
b_wait_inc_full			−0.1742	−0.2500	−0.1785	−0.30
			(0.0929)		(0.0654)	
RMSE	0.1065	0.2988	0.0469	0.0220	0.0545	
					(0.0197)	
MAE	0.0622	0.1751	0.0261	0.0177	0.0391	
					(0.0130)	
MAPE	0.3537	1.4445	0.1121	0.1571	0.2428	
					(0.0706)	

RMSE: Root Mean Squared Error; MAE: Mean Absolute Error; MAPE: Mean Absolute Percentage Error

Why it can be better than the true model? Han et al.'s interpretation

- According to Han et al. (2022)
  - With a wide hidden layer, there is enough flexibility to learn two taste parameters that do not interfere with each other and together optimize the objective function.
  - The introduced L2 regularization constrains the taste parameters, making estimated taste variations relatively stable.

- What we can learn from this?
  - The importance of regularization !
  - Mixed logit model (with the continuous mixture distribution) naturally introduces a L2-like regularization term into the model estimation (see next page).
  - The similar regularization may need to made when the model is fairly complex by introducing interaction terms.

Evgeniou et al. (2007)

- Compare the proposed L2-like regularization method and hierarchical Bayes (HB) for discrete choice model (which is in principle equivalent to the mixed logit (Train, 2009)).
- Confirm that the performance of proposed L2-like regularization method outperforms that of HB through two case studies.

Proposed convex optimization approach

$$\gamma^* = \arg\min_{\gamma} \; cross\text{-}validation(\gamma),$$

$$(\{\mathbf{w}_i^*\}, \mathbf{w}_0^*, D^*) = \arg\min_{\{\mathbf{w}_i\}, \mathbf{w}_0, D} -\frac{1}{\gamma^*}\sum_{i=1}^{I}\sum_{j=1}^{J}\log\frac{e^{\mathbf{x}_{ijq_*}\mathbf{w}_i}}{\sum_{q=1}^{Q} e^{\mathbf{x}_{ijq}\mathbf{w}_i}} + \sum_{i=1}^{I}(\mathbf{w}_i - \mathbf{w}_0)^\top D^{-1}(\mathbf{w}_i - \mathbf{w}_0),$$

subject to D is a positive semidefinite matrix scaled to have trace 1,

HB	L2-like regularization method
Shrinks toward the mean of the first-stage prior	Shrink toward the population mean
Samples from posterior distribution	Minimize a convex loss function
Posterior distribution is a function of parameters of the second-stage priors	Loss function is a function of the regularization parameter γ
The parameters of the second-stage priors are set exogenously	γ is determined using cross-validation

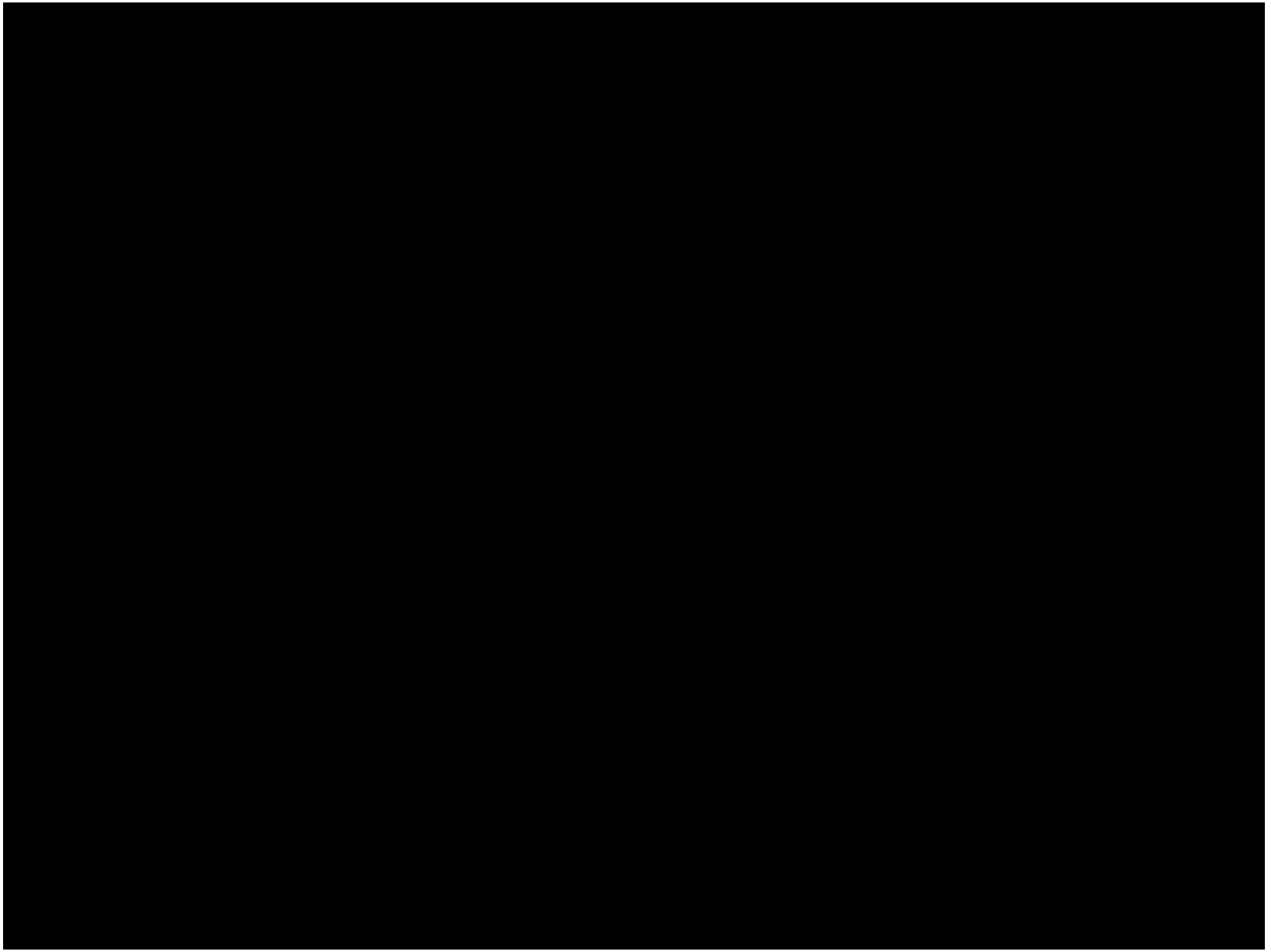
Key would be in the flexibility of the regularization parameter

Take-away messages

- Shifting from “choosing theory-driven OR data-driven” to “integrating theory-driven AND data-driven”.
- To utilize NN in travel behavior analysis, we should design the network structure with paying attention to:
  - How to handle **nonlinearity** in an interpretable manner
  - How to handle **moderation effects** in an interpretable manner
- Han et al.’s (2022) work paying attention to both, while it opens up further possible ways to tune the model.
  - Ishii et al. (2022): Tuning moderation effects using a Tucker decomposition technique.
  - Han et al. avoid positive travel impedance parameters by taking exponential etc., but this does not mean the first derivative is always negative.

References

- Brathwaite, T.: The Holy Trinity: Blending Statistics, Machine Learning and Discrete Choice, with Applications to Strategic Bicycle Planning, PhD Dissertation at UC Berkeley, 2018, https://escholarship.org/uc/item/1pk9p2ct
- Chikaraishi, M., Garg, P., Varghese, V., Yoshizoe, K., Urata, J., Shiomi, Y. and Watanabe, R.: On the possibility of short-term traffic prediction during disaster with machine learning approaches: An exploratory analysis. Transport Policy 98, 91-104, 2020.
- Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H.and Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.
- Cui, Z., Henrickson, K., Ke, R.and Wang, Y.: Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. IEEE Transactions on Intelligent Transportation Systems 21, 4883-4894, 2020.
- Evgeniou, T., Pontil, M., Toubia, O. (2007) A Convex Optimization Approach to Modeling Consumer Heterogeneity in Conjoint Estimation. Marketing Science 26, 805-818.
- Goodfellow I, Bengio Y, Courville A. Deep learning, MIT Press; 2016.
- Han Y, Pereira FC, Ben-Akiva M, Zegras C. A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. Transportation Research Part B: Methodological. 2022;163:166-86.
- Hochreiter, S.and Schmidhuber, J.: Long short-term memory. Neural computation 9, 1735-1780, 1997.
- Ishii, Y., Hayakawa, K., Koide, S., Chikaraishi, M. (2022) Data-driven appraisal of public transport fare policies using an entropy Tucker model with heterogeneous demand (submitted to…).
- Lipton ZC. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue. 2018;16(3):31-57.
- Sifringer, B., Lurkin, V.and Alahi, A.: Enhancing discrete choice models with representation learning. Transportation Research Part B 140, 236-261, 2020.
- Wang, S.: Deep neural networks for choice analysis, PhD Dissertation at MIT,2020 https://dspace.mit.edu/handle/1721.1/129894
- Tong, Z., Liang, Y., Sun, C., Rosenblum, D.S.and Lim, A.: Directed graph convolutional network. arXiv preprint arXiv:2004.13970, 2020. https://arxiv.org/pdf/2004.13970.pdf



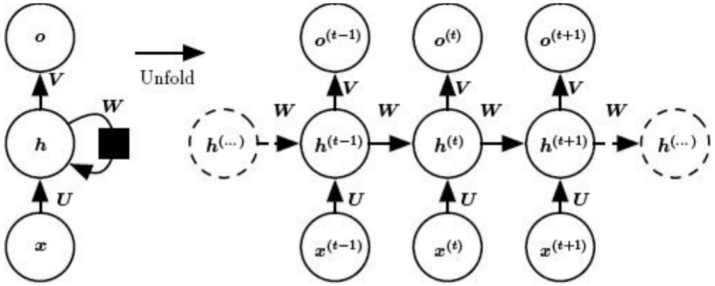
Appendix 1

RECURRENT NEURAL NETWORK

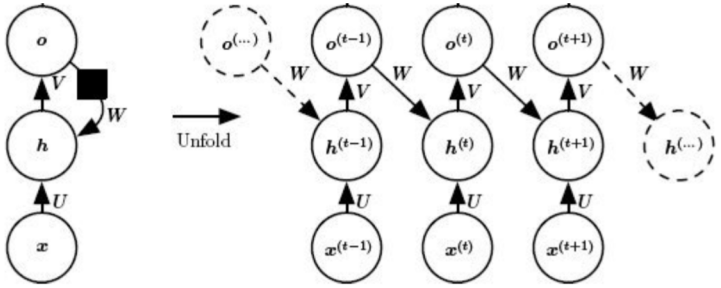
Recurrent neural network

- Recurrent neural network
  - A neural network that is specialized for processing a sequence of values (e.g., time series data).
  - Parameter sharing
    - A recurrent neural network typically shares the same parameters across time steps.
    - This is needed to generalize and make it possible to predict future.
    - An example:
      - Recurrent structure:
        - » Tomorrow will come after today.
      - Non-recurrent structure:
        - » Sep. 18, 2021 will come after Sep. 17, 2021.
  - There are a wide variety of recurrent neural networks (next slide).

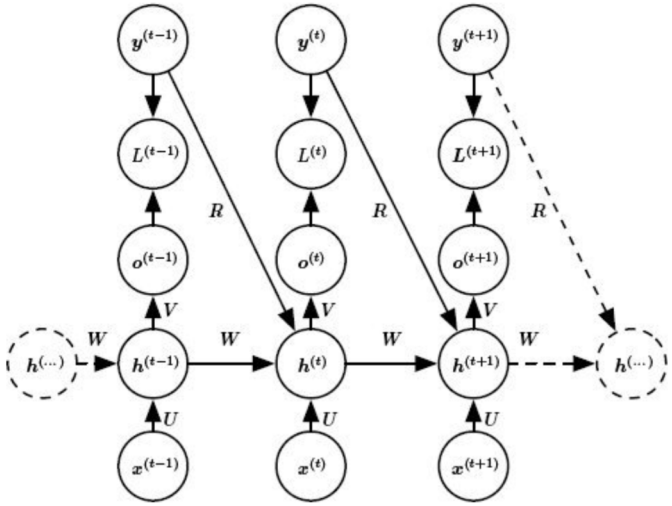
Examples of RNN structures



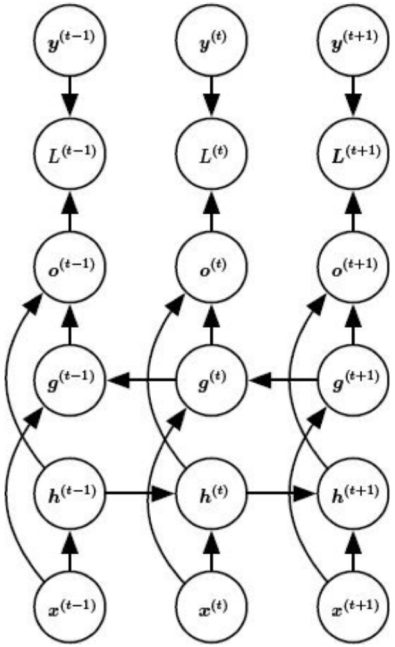
Recurrent networks that produce an output at each time step and have recurrent connections between hidden nodes.



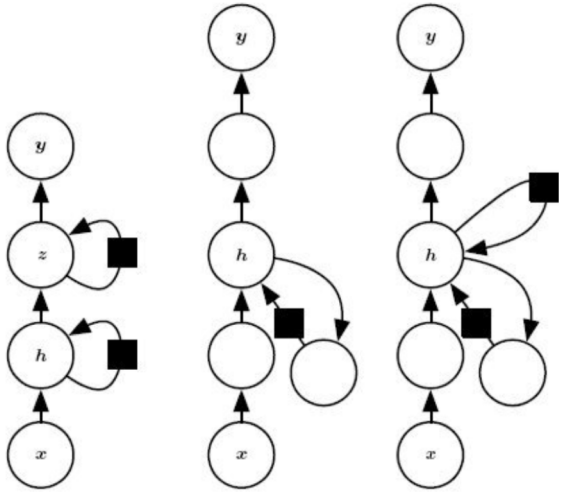
Recurrent networks that produce an output at each time step and have recurrent connections only from the output at the next step to the hidden units at the next time step.



Adding connection from the output at time t to the hidden unit at time t+1



Bidirectional recurrent networks

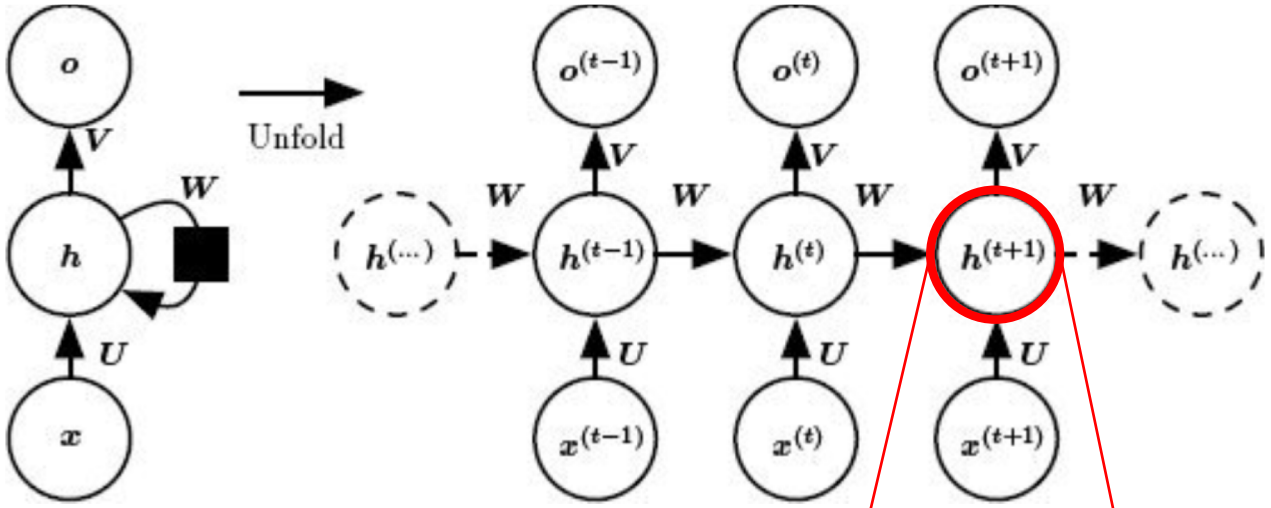


Make network deeper

Various structures exist (similar with time series models with lagged variables)

Hierarchical structure of network: use the concept of “cell”

Having a cell (a set of nodes with a particular network structure), instead of simply having a node.



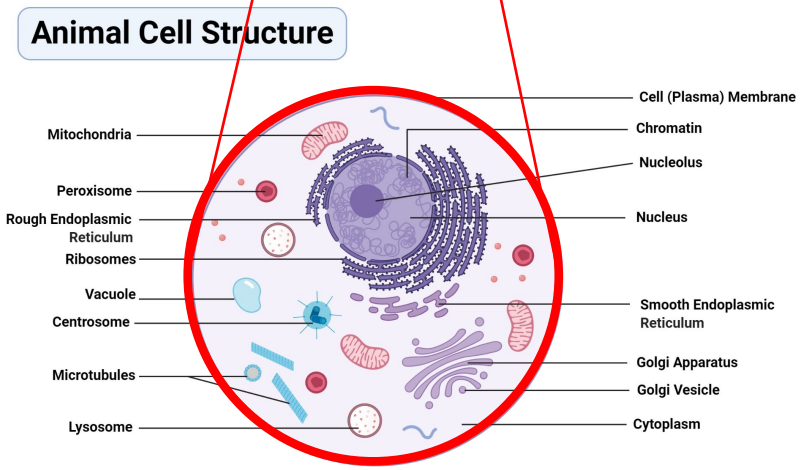


Figure: Animal Cell Structure, Image Copyright © Sagar Aryal, www.microbenotes.com

LSTM cell (Long short-term memory)

Hochreiter and Schmidhuber (1997)

The weight on the self-loop conditioned on the context (rather than fixed), resulting in the dynamic control of the time scale and forgetting behavior of different units.

(Conceptually, it is similar with the introduction of moderators.
Please refer to Yamamoto sensei's lecture note as well)

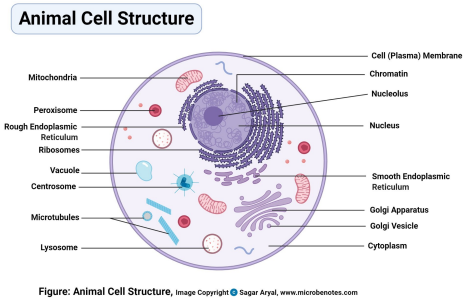
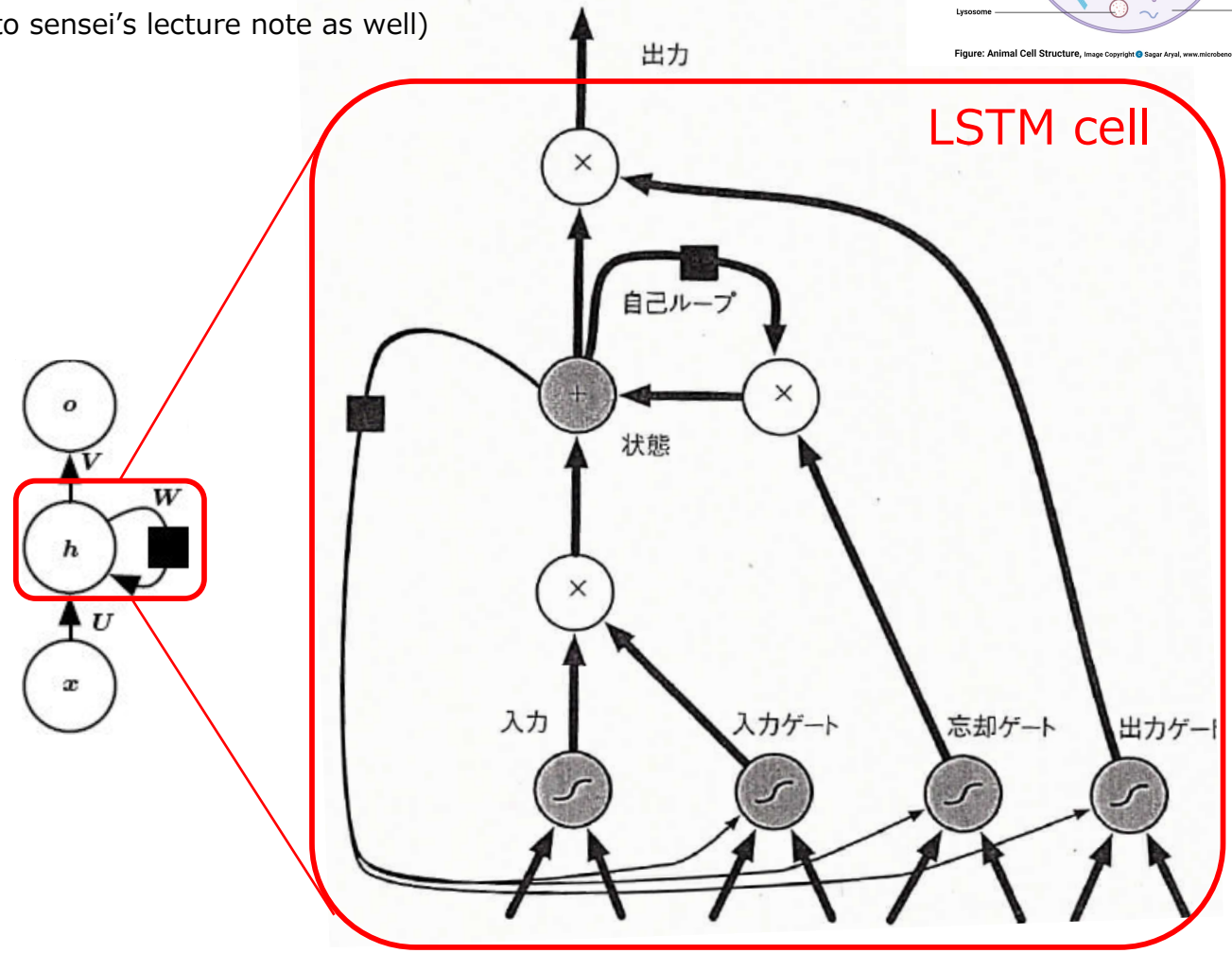
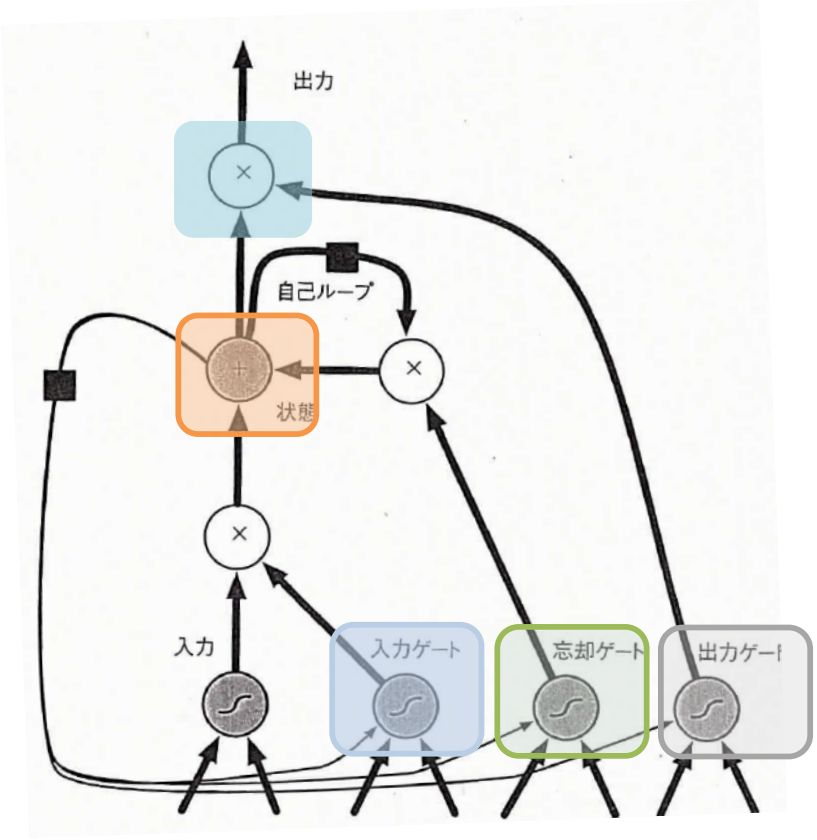


Figure: Animal Cell Structure, Image Copyright Sagar Aryal, www.microbenotes.com



LSTM cell (Long short-term memory)

Hochreiter and Schmidhuber (1997)



Forget gate

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_i^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

Input gate

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_i^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

State

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

Output

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}$$

Output gate

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_i^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right)$$

Other variants...

- ✓ GRU: Gated recurrent unit (Cho et al., 2014)
  - ✓ Simpler than LSTM, but the performance is similar to that of LSTM for some applications.