A short tutorial of deep learning and computational graph frameworks in transportation modeling

Method · May 2018

DOI: 10.13140/RG.2.2.26214.32321

2 authors:

Xin Wu
Beijing Jiaotong University

Xuesong Simon Zhou
Arizona State University

A short tutorial of deep learning and computational graph frameworks in transportation modeling

Prepared by Xin Wu and Xuesong Zhou

In this tutorial, we provide an interesting illustrative example, adapted from Koppelman and Bhat (2006) in the context of transportation demand modeling, to show the relationship between the multinomial logit model and a three-layer artificial neural network and computational graph frameworks. Consider a case with two available alternatives: Drive Alone (DA) and Transit (TR). The following utility function shows that the decision-maker preferences are a function of income (INC) and travel time (TT). For simplicity, we do not consider interaction terms and constant terms.

$$U_{DA} = \beta_{DA,1} INC - \beta_2 TT_{DA} \tag{1}$$

$$U_{TR} = \beta_{TR,1} INC - \beta_2 TT_{TR} \tag{2}$$

Then, one can calculate the probability of choosing DA using the following multinomial logit model:

$$P_{DA} = \frac{\exp(U_{DA})}{\exp(U_{TR}) + \exp(U_{DA})} = \frac{1}{1 + \exp(U_{TR} - U_{DA})} \tag{3}$$

Table 1 shows an example with $\beta_{DA,1} = 0.004$, $\beta_{TR,1} = 0$ and $\beta_2 = 0.02$, for an individual from a household with a 50-thousand-dollar annual income and facing travel times of 30 and 50 minutes for DA and TR, respectively. The utility and probability calculation are shown in Table 1.

Table 1. Utility and Probability Calculation with Transit as Base Alternative

Alternative	Utility		Exponent	Probability
	Expression	Value		
Drive alone	$U_{DA} = \beta_{DA,1} INC - \beta_2 TT_{DA} = 0.004 \times 50 - 0.02 \times 30$	-0.4	0.6703	$P_{DA} = 0.65$
Transit	$U_{TR} = \beta_{TR,1} INC - \beta_2 TT_{TR} = 0 \times 50 - 0.02 \times 50$	-1	0.3679	$P_{TR} = 0.35$
	$\beta_{DA,1} = 0.004$; $\beta_{TR,1} = 0$; $\beta_2 = -0.02$		$\sum = 1.0382$	

Then, the multinomial logit model for binary mode choice can be viewed as the simplest neural network with only three layers corresponding to two steps of calculations. The first layer is a stack of neurons that express the linear utility function of the differences between pairs of alternatives for prediction of DA probability.

$$U_{TR} - U_{DA} = (\beta_{TR,1} - \beta_{DA,1}) INC - \beta_2 (TT_{TR} - TT_{DA}) \tag{4}$$

Consider $-\beta_1 = \beta_{TR,1} - \beta_{DA,1}$, then

$$U_{TR} - U_{DA} = -\beta_1 INC - \beta_2 (TT_{TR} - TT_{DA}) = -\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR} \tag{5}$$

where $\boldsymbol{x} = (-INC, TT_{DA} - TT_{TR})$ can be viewed as a vector including both income and travel time. The second layer applies the logistic sigmoid function $\sigma(U_{DA} - U_{TR})$ as an activation function to squeeze the output of the linear utility function into the interval (0, 1). The third layer indicates the choice probability of DA P_{DA}:

$$P_{DA} = \sigma(U_{DA} - U_{TR}) = \frac{1}{1 + \exp(U_{TR} - U_{DA})} = \frac{1}{1 + \exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR})} \tag{6}$$

Fig. 1 shows two styles of neural networks that express the multinomial logit model. The multinomial logit model is a composite function consisting of both a linear utility function Eq. (5) and sigmoid function Eq. (6).

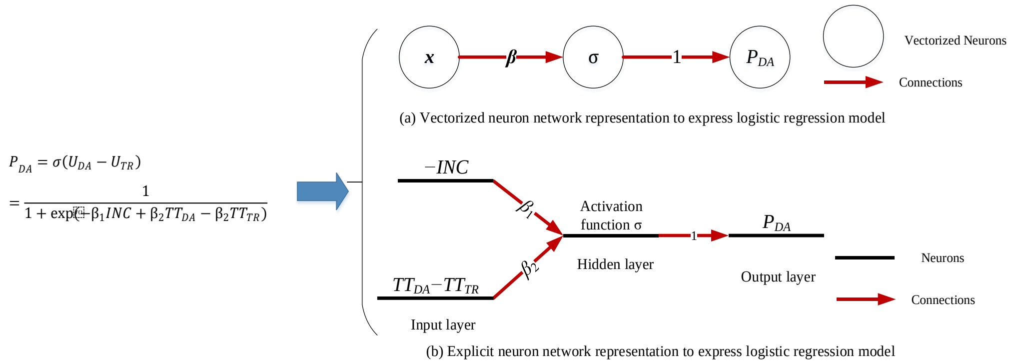


Fig. 1. An illustrative example to show the relationship between multinomial logit and a three-layer ANN.

As an illustrative example based on the neural network in Fig.1, Fig.2 shows how to describe function $P_{DA} = \sigma(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR})$ using a computational graph. There are seven operations in the expression including two additions, three multiplications, one exponential operation, and one reciprocal operation. As shown in Fig.2, we can extend the three-layer ANN in a computational graph for the expression using inputs, outputs and the following six intermediary variables as vertexes:

$$a = -\beta_1 INC \tag{7}$$

$$b = -\beta_2 TT_{TR} \tag{8}$$

$$c = \beta_2 TT_{DR} \tag{9}$$

$$u = a + b + c = -\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR} \tag{10}$$

$$e = \exp(u) = \exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR}) \tag{11}$$

$$d = u + 1 = \exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR}) + 1 \tag{12}$$

We evaluate the output value of P_{DA} by setting the input variables/parameters to certain values and then computing the vertexes up through the graph. Let us set $\beta_1 = 0.004$ and $\beta_2 = 0.02$, $INC = 50$ thousand dollar, $TT_{DA} = 30$ minutes, and $TT_{TR} = 50$ minutes. We obtain results, as same as Table a1, using the computational graph ($P_{DA} = 0.65$).

In the computational graph plotted in Fig. 2, each edge corresponds to a derivative. For example, edge ① corresponds to the partial derivative of P_{DA} with respect to d and edge ② corresponds to the partial derivative of d with respect to e. To understand partial derivatives in these cases, it is important to understand the chain rule in calculus. According to the chain rule, to calculate the partial derivatives of P_{DA} with respect to any parameter, we need to sum up all possible paths from vertex P_{DA} to the vertexes of the variables/parameters in the graph, and multiply the derivatives on each edge of the path together. To avoid duplicated calculations, a dynamic programming principle can be applied to update the derivatives sequentially by following the chain rules along the "computing" paths in the computational graph in a backwards fashion.

(i)$\frac{\partial P_{DA}}{\partial \beta_1}$ can be calculated by multiplying the partial derivatives on path ①→②→③→⑥→⑦.

$$\frac{\partial P_{DA}}{\partial \beta_1} = \frac{\partial P_{DA}}{\partial d}\frac{\partial d}{\partial e}\frac{\partial e}{\partial u}\frac{\partial u}{\partial a}\frac{\partial a}{\partial \beta_1} = \left(-\frac{1}{d^2}\right)\exp(u)\,(-INC) = \frac{INC\,\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR})}{[\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR}) + 1]^2} \tag{13}$$

(ii) $\frac{\partial P_{DA}}{\partial \beta_2}$ can be calculated by summing up the multiplied partial derivatives on path ①→②→③→④→⑨ and the multiplied partial derivatives on path ①→②→③→⑤→⑧.

$$\frac{\partial P_{DA}}{\partial \beta_2} = \frac{\partial P_{DA}}{\partial d}\frac{\partial d}{\partial e}\frac{\partial e}{\partial u}\frac{\partial u}{\partial b}\frac{\partial b}{\partial \beta_2} + \frac{\partial P_{DA}}{\partial d}\frac{\partial d}{\partial e}\frac{\partial e}{\partial u}\frac{\partial u}{\partial c}\frac{\partial c}{\partial \beta_2} = \left(-\frac{1}{d^2}\right)\exp(u)\,(TT_{DA} - TT_{TR}) =$$
$$= \frac{(TT_{TR} - TT_{DA})\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR})}{[\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR}) + 1]^2} \tag{14}$$

For all three paths, we can calculate

$$\frac{\partial P_{DA}}{\partial d}\frac{\partial d}{\partial e}\frac{\partial e}{\partial u} = -\frac{\exp(u)}{d^2} = -0.4162 \times 1 \times 0.55 = -0.2284 \tag{15}$$

Then using the principle of dynamic programming

$$\frac{\partial P_{DA}}{\partial \beta_1} = -0.2284 \times (-50 \times 1) = 11.4217 \tag{16}$$

$$\frac{\partial P_{DA}}{\partial \beta_2} = -0.2284 \times (30 - 50) = 4.5687 \tag{17}$$

It should be highlighted that above partial derivatives calculated using the computational graph is as same as the formulations used in discrete choice models (Koppelman and Bhat, chapter 4, 2006). That is

$$\frac{\partial P_{DA}}{\partial \beta_1} == \frac{INC\,\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR})}{[\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR}) + 1]^2} = INC\,\sigma(U_{DA} - U_{TR})[1 - \sigma(U_{DA} - U_{TR})] \tag{18}$$

$$\frac{\partial P_{DA}}{\partial \beta_2} == \frac{(TT_{TR} - TT_{DA})\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR})}{[\exp(-\beta_1 INC + \beta_2 TT_{DA} - \beta_2 TT_{TR}) + 1]^2}$$
$$= (TT_{TR} - TT_{DA})\,\sigma(U_{DA} - U_{TR})[1 - \sigma(U_{DA} - U_{TR})] \tag{19}$$

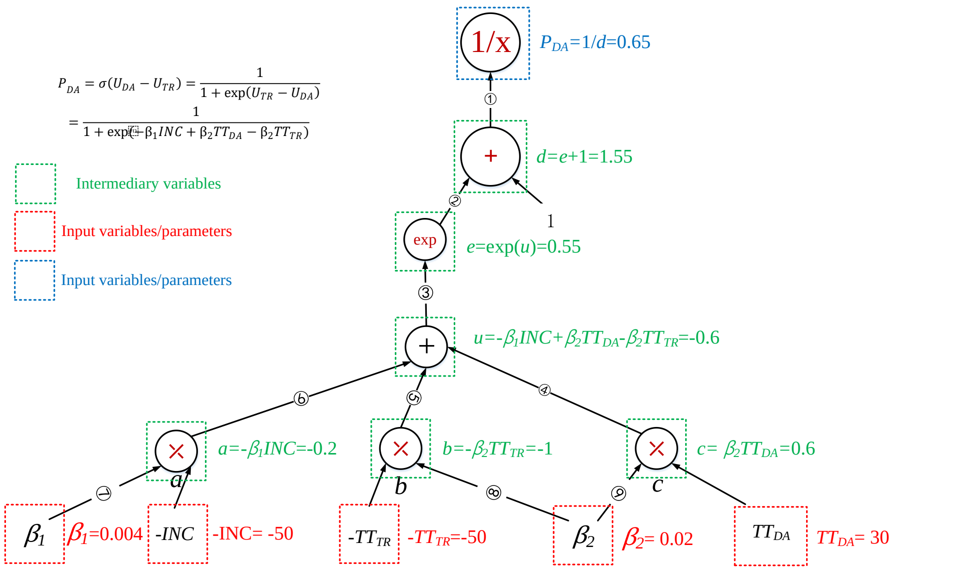
Typically, the estimation "loss errors" can be propagated to update estimation variables, e.g., β_1, β_2 , across the computational graph using the stochastic gradient descent (SGD) algorithm. This minimizes the sum of Euclidean residuals for a set of individual observations labeled $\overline{\boldsymbol{x}}^m = (\overline{INC}^m, \overline{TT}_{DA}^m, \overline{TT}_{TR}^m)$ and binary sample $\bar{y}^m \in \{0,1\}$, where $m = 1,2,\dots,M$:

$$F(\beta_1, \beta_1) = \min_{\beta_1,\beta_2} \frac{1}{2}\sum_{m=1}^{M}(\bar{P}_{DA}^m - \bar{y}^m)^2 \tag{20}$$

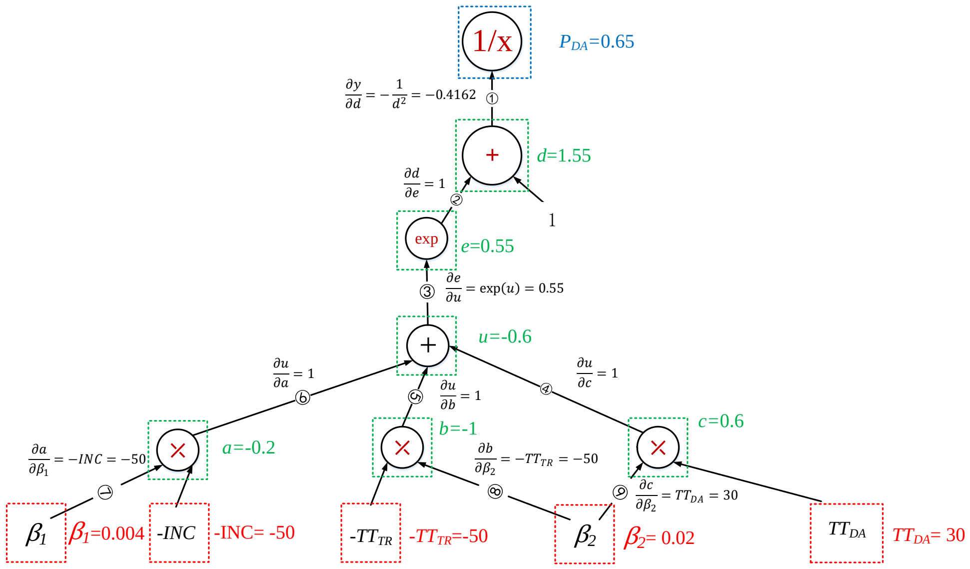
The gradients $\frac{\partial F(\beta_1,\beta_1)}{\partial \beta_1}$ and $\frac{\partial F(\beta_1,\beta_1)}{\partial \beta_2}$ can then be calculated easily. That is

$$\frac{\partial F(\beta_1,\beta_1)}{\partial \beta_1} = \sum_{m=1}^{M} \frac{\partial F(\beta_1,\beta_1)}{\partial \bar{P}_{DA}^m} \frac{\partial \bar{P}_{DA}^m}{\beta_1}; \qquad \frac{\partial F(\beta_1,\beta_1)}{\partial \beta_2} = \sum_{m=1}^{M} \frac{\partial F(\beta_1,\beta_1)}{\partial \bar{P}_{DA}^m} \frac{\partial \bar{P}_{DA}^m}{\partial \beta_2} \tag{21}$$

where $\frac{\partial F(\beta_1,\beta_1)}{\partial\, \partial \bar{P}_{DA}^m} = \bar{P}_{DA}^m - \bar{y}^m$.



(a) Expression calculation using the computational graph



(b) Derivatives on the computational graph

Fig. 2. An illustrative computational graph for a multinomial logit model for function $\sigma(U_{DA} - U_{TR})$

[1]. Koppelman, F. S., Bhat, C., 2006. A self instructing course in mode choice modeling: multinomial and nested logit models. US Department of Transportation, Federal Transit Administration, 31. Doi: https://doi.org/10.1002/stem.294