Understanding Computational Graph

Xuesong (Simon) Zhou, xzhou74@asu.edu

School of Sustainable Engineering and the Built Environment
Arizona State University

Co-author: Dr. Xin (Bruce) Wu
xinwu3@asu.edu
Postdoc scholar at ASU

Arizona State University

Contents

1. What is **Deep Learning** from our model calibration perspective?

2. What is **layered computation graph?**

Arizona State University

1. What is Deep Learning from our model calibration perspective?

What can we learn from AI field?

1. Hierarchical structure of artificial neural networks

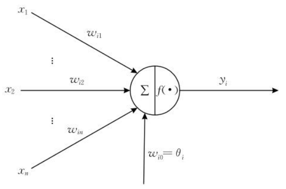
Brief history of deep learning

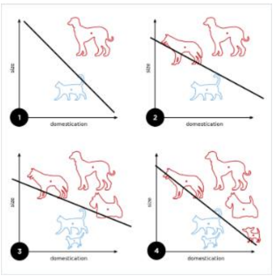
McCulloch-Pitts (1943)
Neuron of brain function

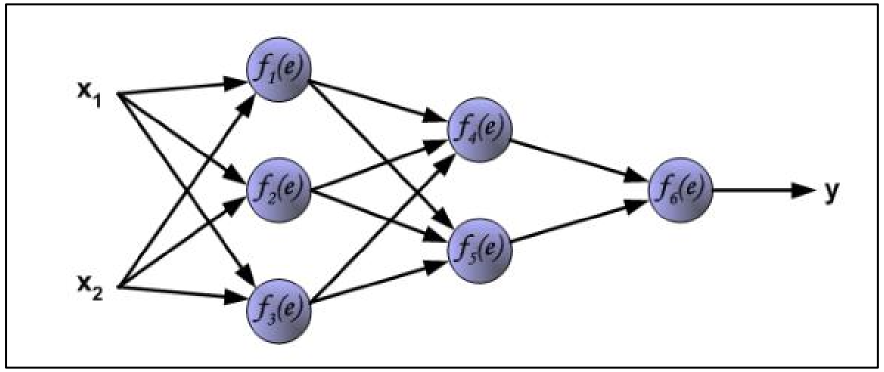
Rosenblatt (1956)
Perceptron ADA-LINE
Stochastic gradient descent

Rumelhart et al. (1986)
Multi-layer distributed representation
Back propagation algorithm

Hinton et al. (2006)
The last ten years
Convolutional networks







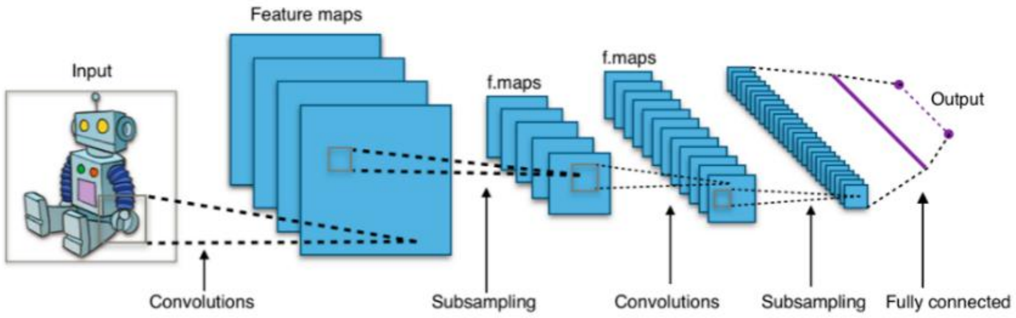
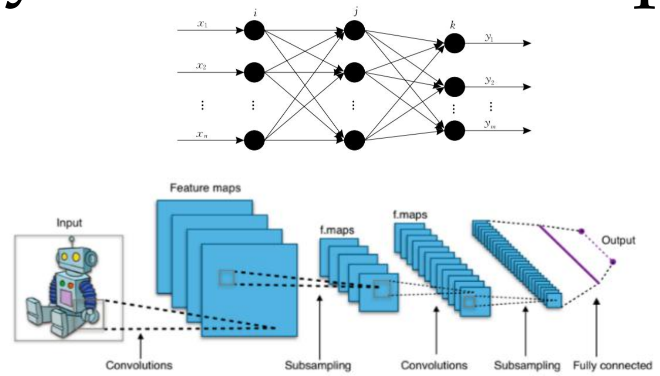
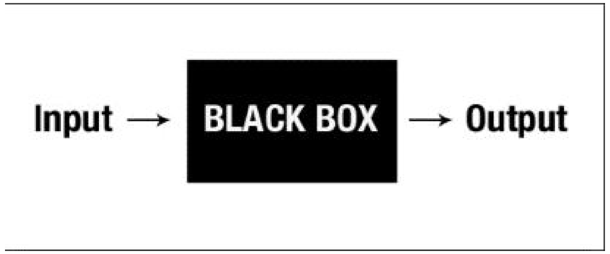


Figure source: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

What can we learn from AI field?

Why cannot use deep learning directly?





Do not have Interpretability

Ground truth

3000 households; 3.333 trips/h

Number of trips: 10,000

66.15%: 6,615 trips on the freeway

33.85%: 3,385 trips on the arterial

Current toll: 1 dollar

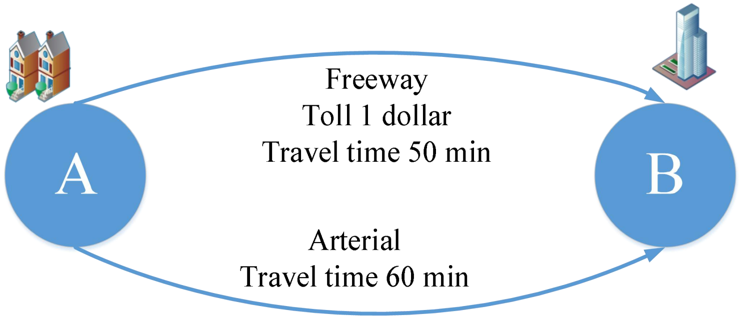
VOT:10 dollar/h

Real-world travel time of freeway: 50 min

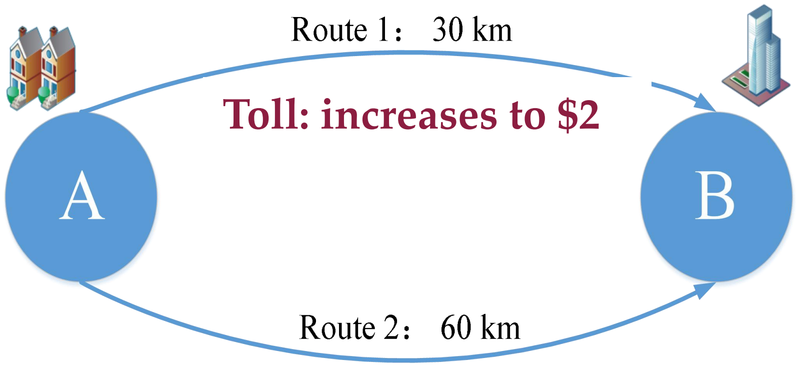
Real-world travel time of arterial: 60 min

If the toll increases to 2 dollars, how the demand will changes?

Now



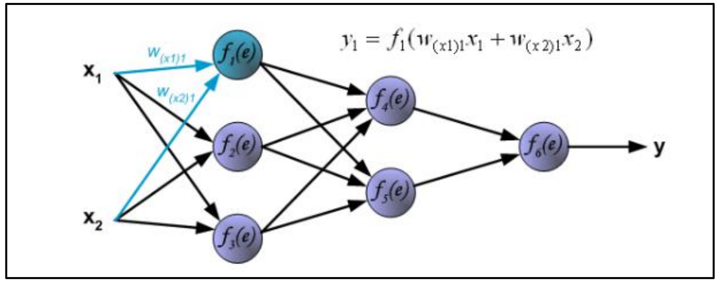
Future

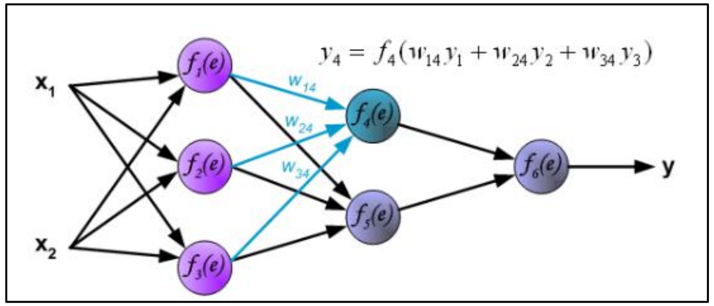


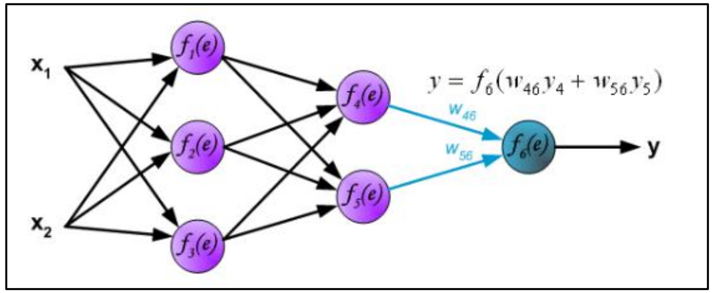
What can we learn from AI field?

2. Forward and Backward propagation

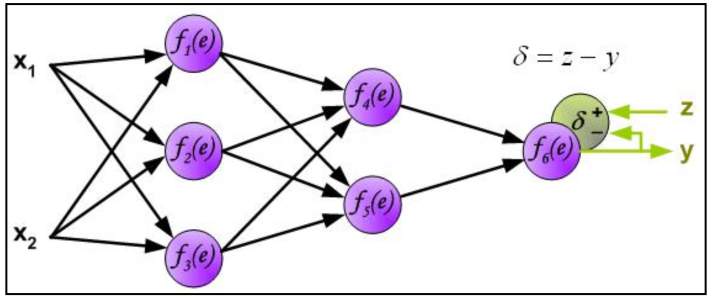
Forward propagation

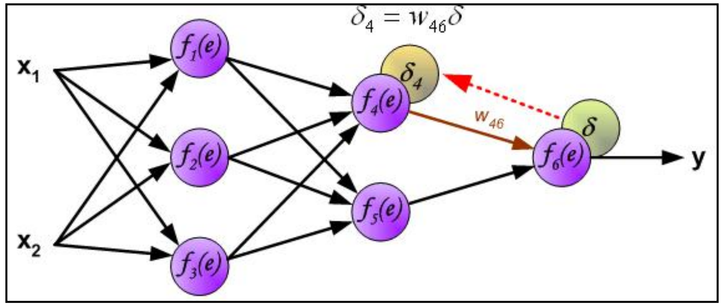


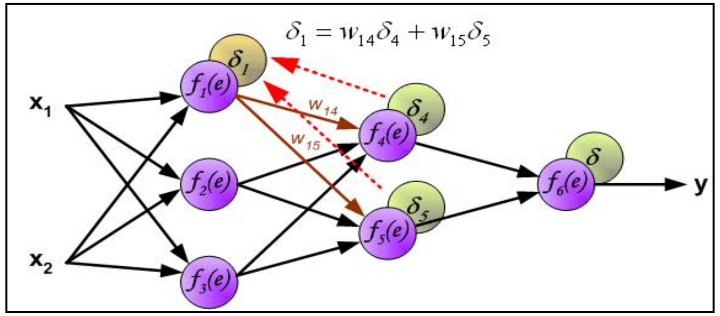




Backward propagation







Arizona State University

Source: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

What can we learn from AI field?

Math foundation of neural network architecture:

Computational graph approach

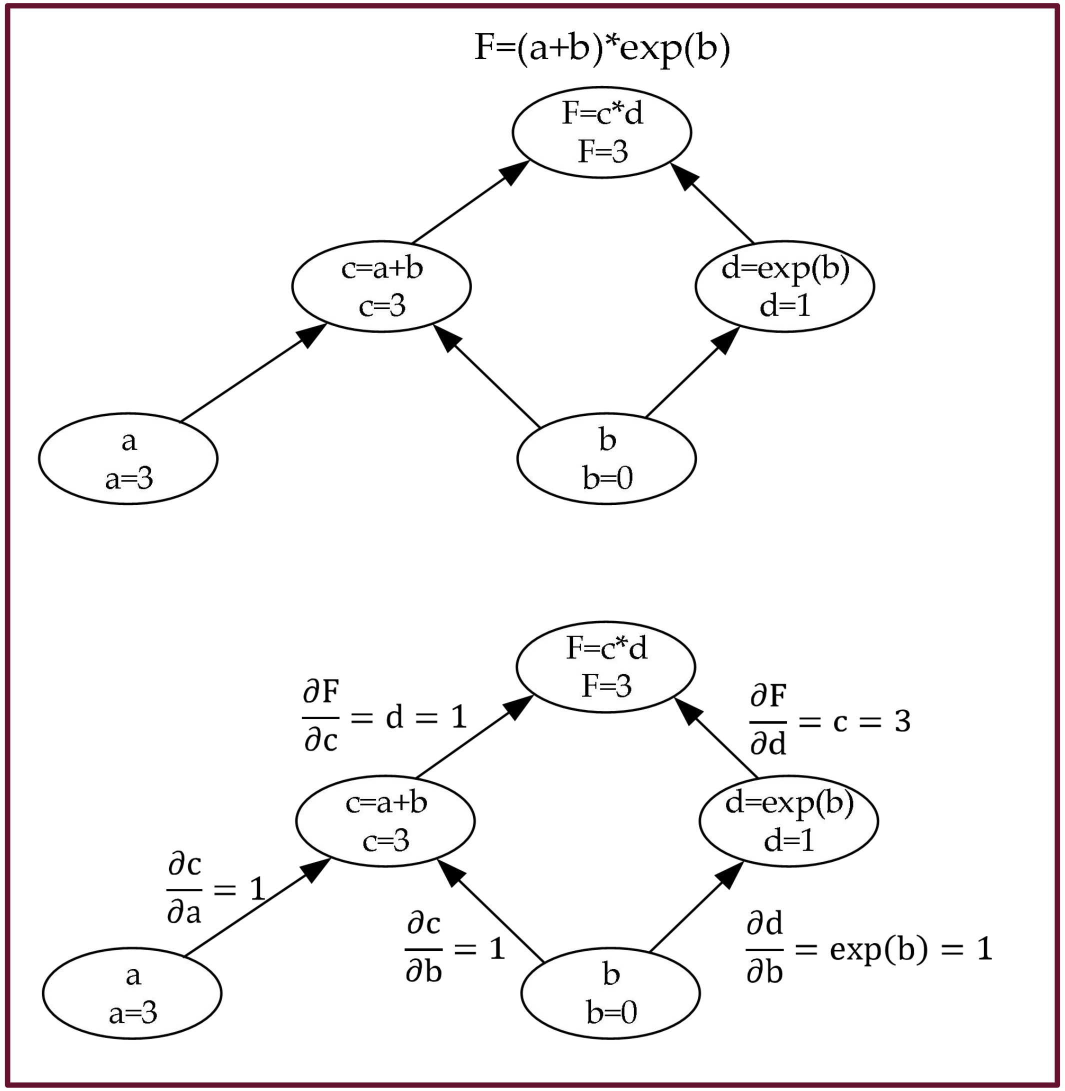
- Computational graph is a acyclic graph to express composite mathematical formulations
  (A generalization of Artificial Neural Network)

- Computational graph is a technique for calculating derivatives quickly
  (A generalization of Back propagation algorithm)

To evaluate the partial derivatives in this graph, we just need to "summing over the paths". For example, to get the derivative of **F** with respect to **b** by:

$$\frac{\partial F}{\partial b} = \frac{\partial F}{\partial c}\frac{\partial c}{\partial b} + \frac{\partial F}{\partial d}\frac{\partial d}{\partial b} = 1 * 1 + 3 * 1 = 4$$

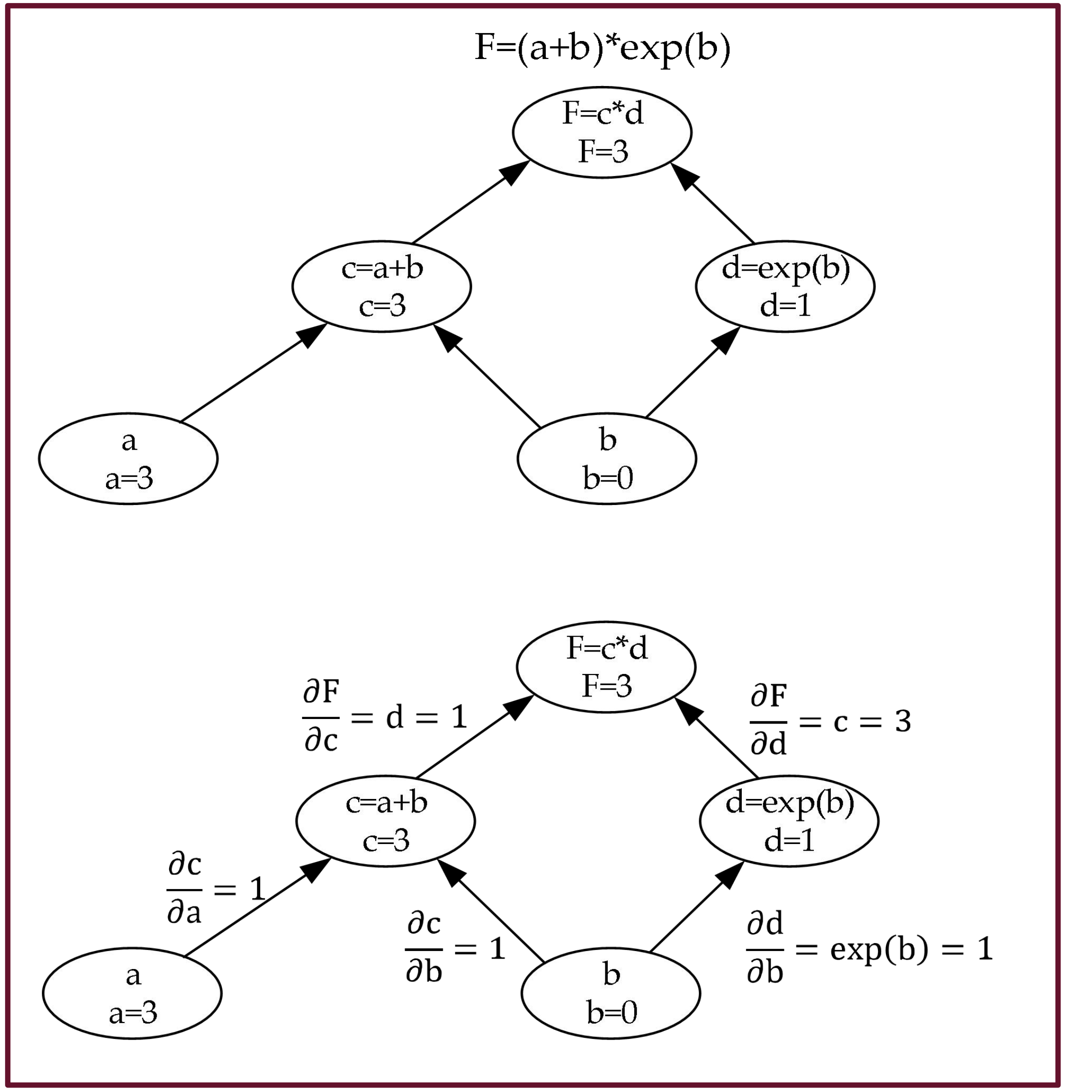
$$\frac{\partial F}{\partial a} = \frac{\partial F}{\partial c}\frac{\partial c}{\partial a} = 1 * 1 = 1$$



Adapted from : http://colah.github.io/posts/2015-08-Backprop/

Tensorflow sample code

```
import tensorflow as tf
import pandas as pd
import numpy as np
from datetime import datetime


# In[33]:


# Define constant values
a = tf.constant(3, dtype=np.float32)
b = tf.constant(0, dtype=np.float32)


# In[34]:


# Define the mathematical formula
@tf.function
def model_fun(a, b):

    F = tf.math.multiply(tf.math.add(a,b), tf.exp(b))


    return F

Result = model_fun(a, b)
print (Result)
```



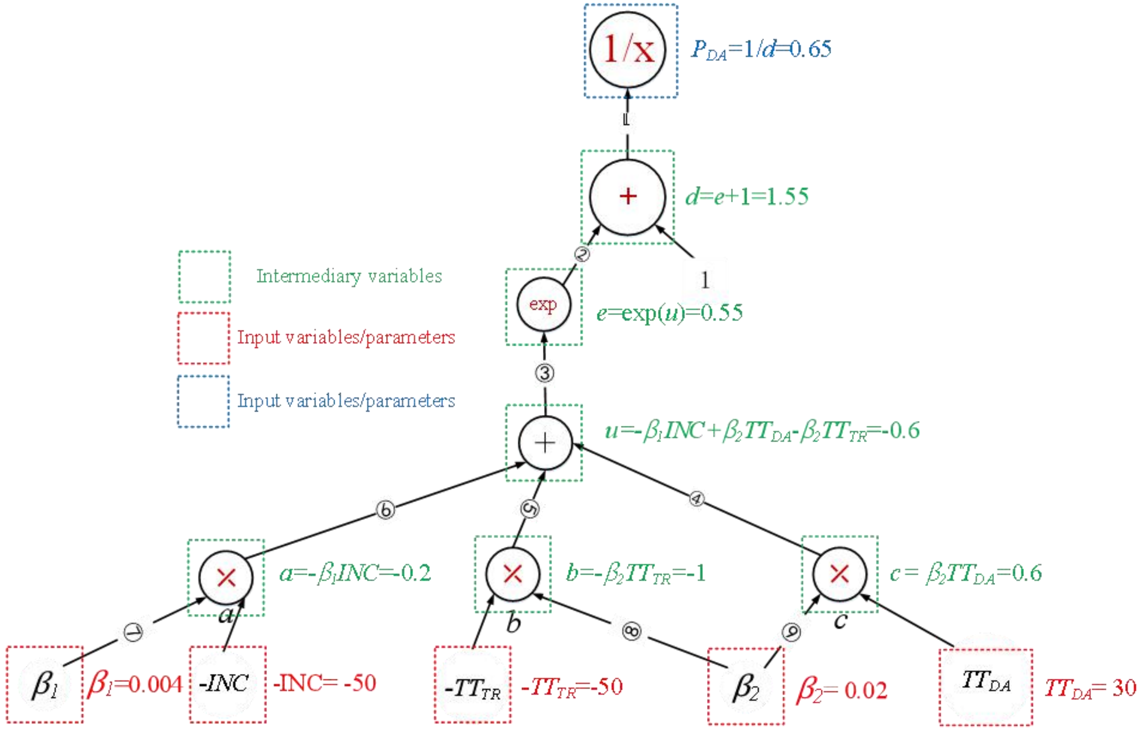
What can we learn from AI field?

3. One foundation for Deep Learning: Computational graph

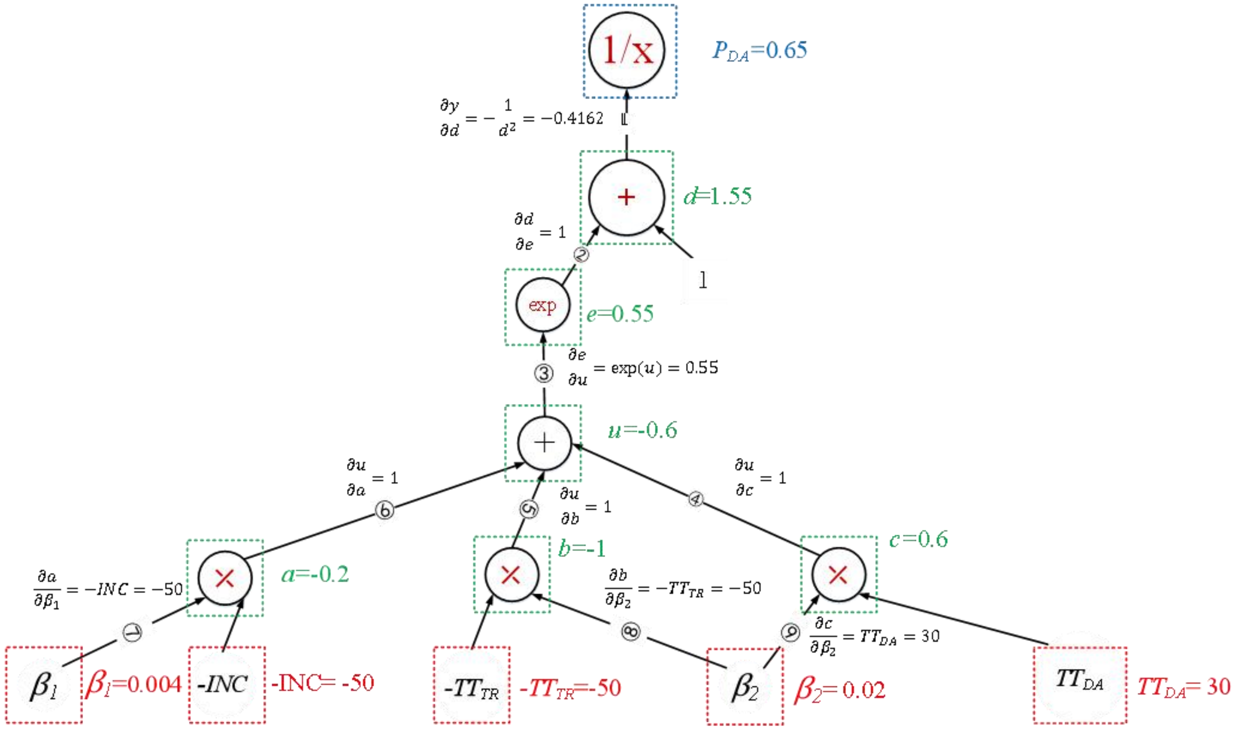
Mode split models (driving alone vs. transit service) **CAN ALSO BE EXPRESSED** by a computational graph

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

$$\frac{\partial P_{DA}}{\partial \beta_2} = \frac{\partial P_{DA}}{\partial d}\frac{\partial d}{\partial e}\frac{\partial e}{\partial u}\frac{\partial u}{\partial b}\frac{\partial b}{\partial \beta_2} + \frac{\partial P_{DA}}{\partial d}\frac{\partial d}{\partial e}\frac{\partial e}{\partial u}\frac{\partial u}{\partial c}\frac{\partial c}{\partial \beta_2} = (TT_{TR} - TT_{DA})\,\sigma(U_{DA} - U_{TR})[1 - \sigma(U_{DA} - U_{TR})]$$



(a) Expression calculation using the computational graph



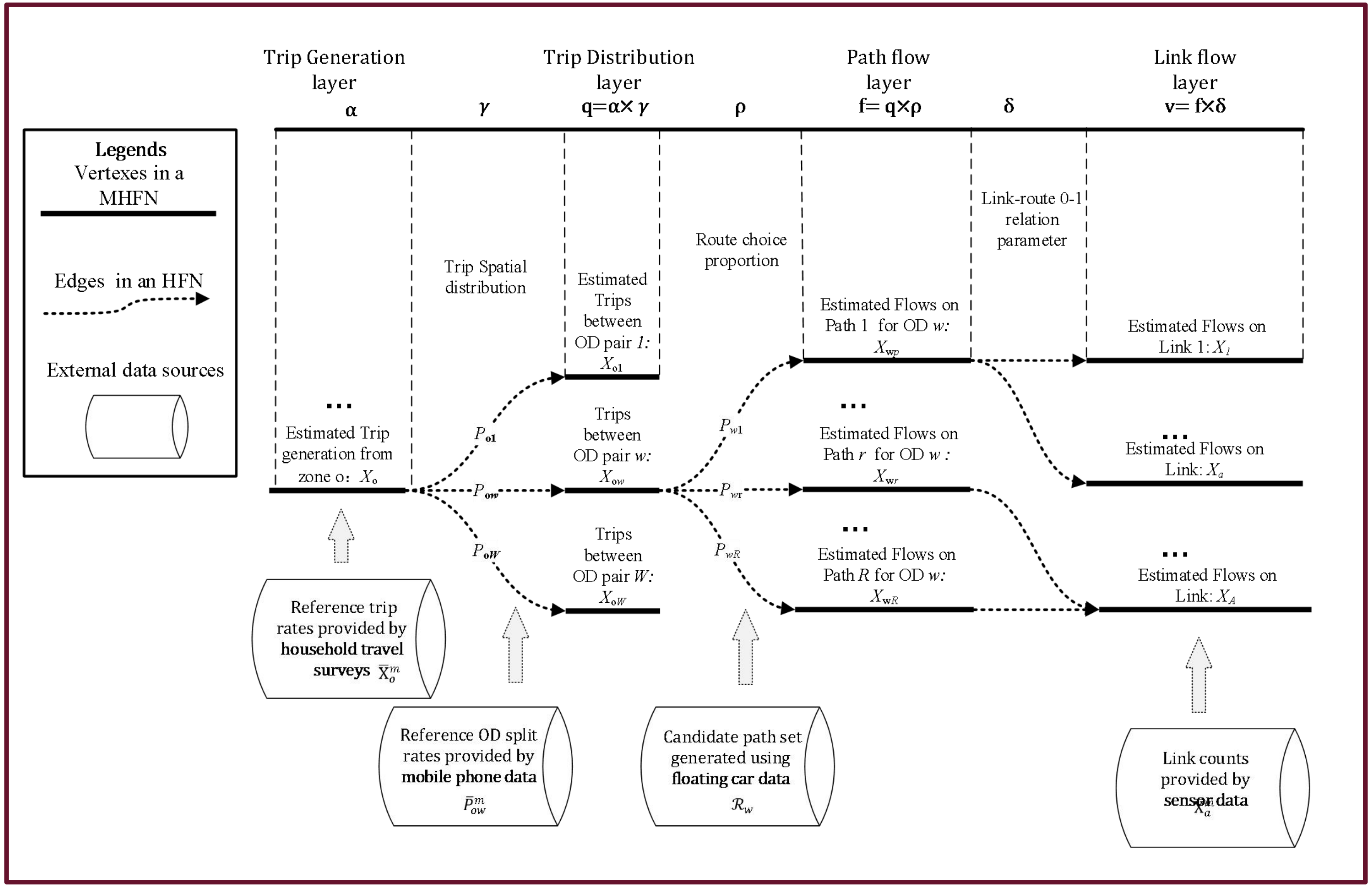
(b) Derivatives on the computational graph

2.What is layered computation graph?

A forward and backward propagation algorithmic framework on a layered computational graph

2.1 Hierarchical Flow Network (HFN) representation

HFN representation in a explicit form

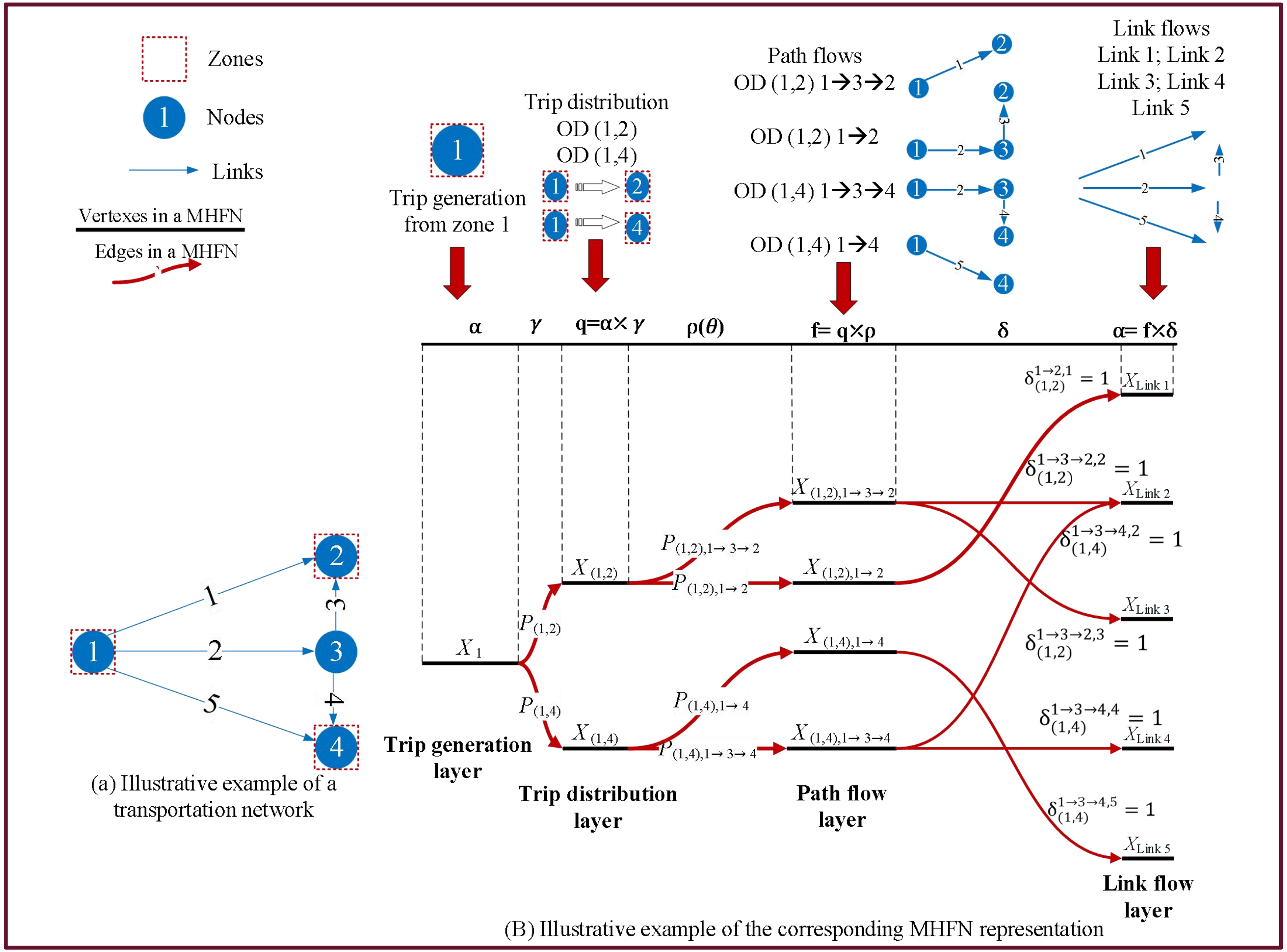


1. **The following flow conservation constraints are expressed by the "Neural Network" whose activation functions are ReLu function** $f(x) = \max(0, x)$:

$$X_o P_{ow} = X_w \;\; \forall\, w \in \mathcal{W} \; o \in \mathcal{Z}$$
$$X_w P_{wr} = X_r \; \forall\, r \in \mathcal{R} \; w \in \mathcal{W}$$
$$\sum_{r \in \mathcal{R}} \delta_{ra} X_r = X_a \;\; \forall a \in \mathcal{A}$$

2. Different types of data sources are mapped onto different layers of the architecture

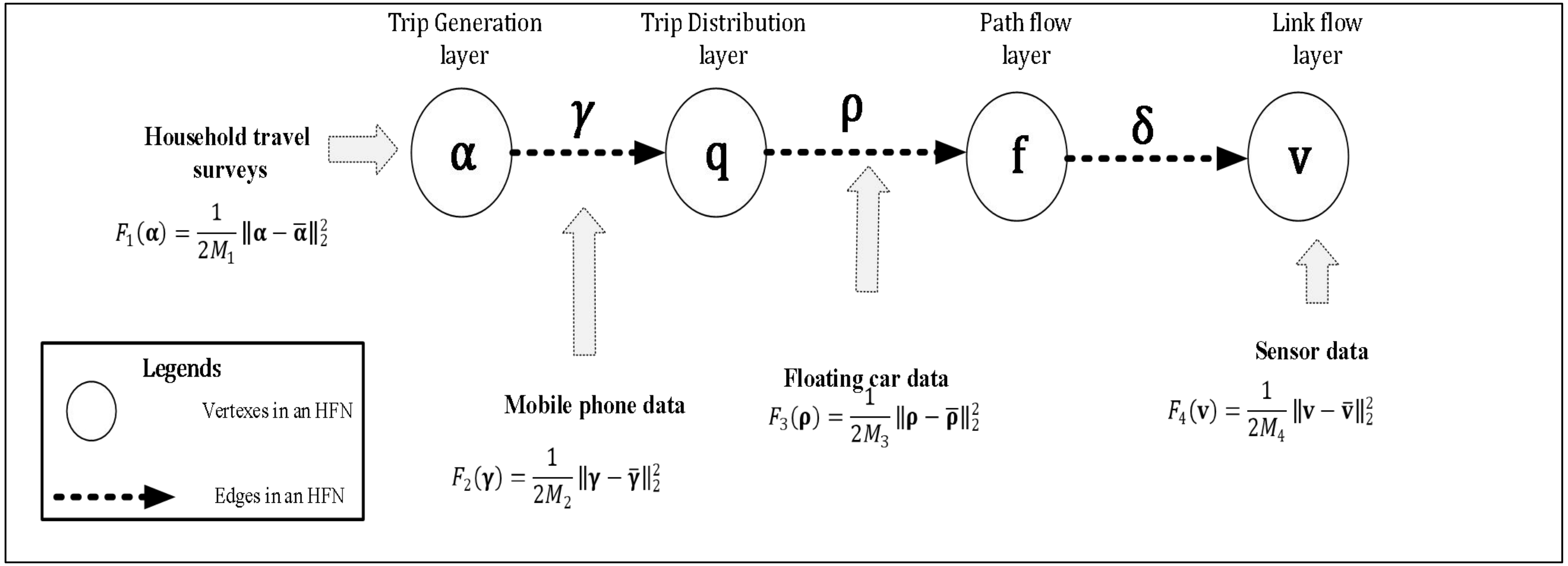
Example of an HFN



(a) Illustrative example of a transportation network

(B) Illustrative example of the corresponding MHFN representation

HFN representation in a vectorized form



1. Different types of data sources generate different loss function:

$$F_1(\boldsymbol{\alpha}) = \frac{1}{2M_1}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \text{ (Survey)}$$

$$F_2(\boldsymbol{\gamma}) = \frac{1}{2M_2}\|\boldsymbol{\gamma} - \bar{\boldsymbol{\gamma}}\|_2^2 \text{ (Phone)}$$

$$F_3(\boldsymbol{\rho}) = \frac{1}{2M_3}\|\boldsymbol{\rho} - \bar{\boldsymbol{\rho}}\|_2^2 \text{ (GPS)}$$

$$F_4(\boldsymbol{v}) = \frac{1}{2M_4}\|\boldsymbol{v} - \bar{\boldsymbol{v}}\|_2^2 \text{ (Senor)}$$

$$\boldsymbol{\rho}_r = \frac{exp(U_r)}{\sum_{p\in \boldsymbol{P}(w)} exp(U_r)}$$

$$U_r = -\beta_{w,1}TC_r - \beta_{w,2}TT_r + \beta_w$$

Travel time of path r: TT_r
Travel cost of path r: TC_r

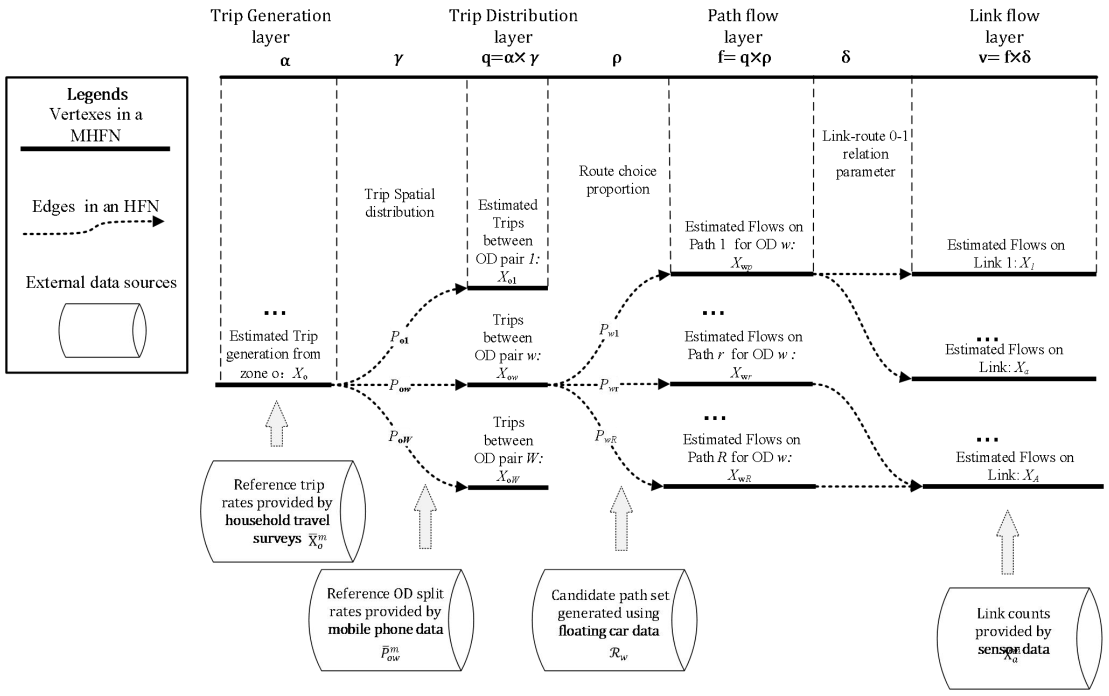
Multinomial logit model
(also known as **SOFTMAX function** in deep learning field)
In short
$\boldsymbol{\rho} = \text{softmax}(\boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \boldsymbol{\beta}, \mathbf{TC}, \mathbf{TT})$

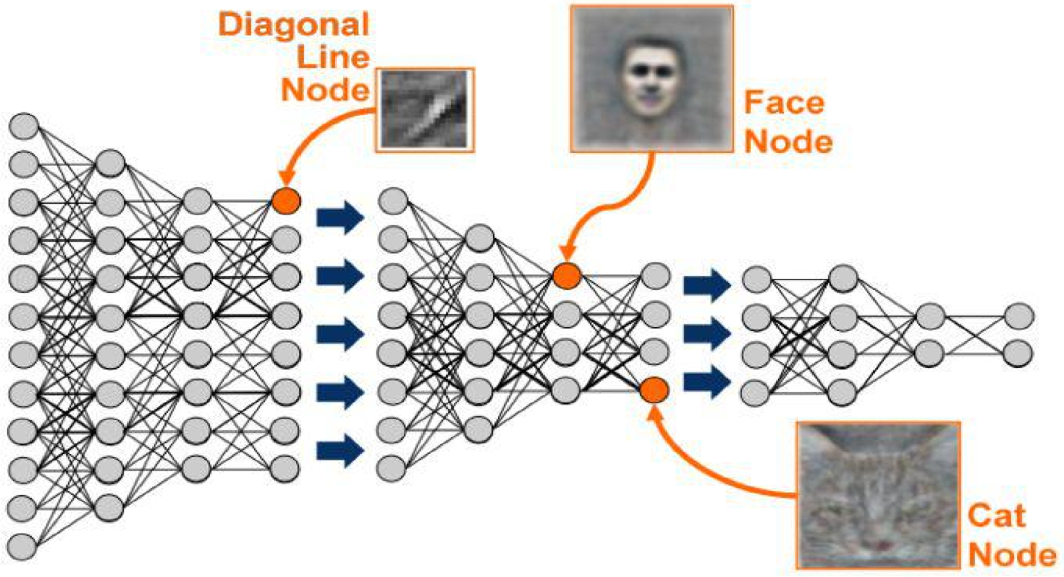
2. The 3 steps of the 4-step process can be expressed:

$$\boldsymbol{\alpha} \times \boldsymbol{\gamma} = \mathbf{q}$$
$$\mathbf{q} \times \boldsymbol{\rho} = \mathbf{f}$$
$$\mathbf{f} \times \boldsymbol{\delta} = \mathbf{v}$$
$$\boldsymbol{\rho} = \text{softmax}(\boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \boldsymbol{\beta}, \mathbf{TC}, \mathbf{TT})$$

where $\boldsymbol{\alpha}$, $\mathbf{q}$, $\mathbf{f} \geq 0$; $\boldsymbol{\gamma}$, $\boldsymbol{\rho} \in [\mathbf{0}, \mathbf{1}]$
$\boldsymbol{\delta}$ is the path-link incident matrix

Compare HFN with Artificial Neural Networks





Similarity:

- Hierarchy of the model
- Transitivity (Error signals are
- Forward and backward propagated on the network)

Differences:

- Data are input on different layers in HFNs
- Each "Neuron" has explainable traffic meaning

■ Backpropagation Algorithm

- **History**
- **Backpropagation** is a method to calculate the gradient of the loss function with respect to the weights in an artificial neural network.
- **Backpropagation** were derived in the context of control theory by Herry K. Kelley in 1960 and by Arthur E. Bryson in 1961, using principles of **dynamic programming**.
- **Intuition**
- Learning as an **optimization problem**.
- **Limitations**
- Gradient descent with backpropagation is not guaranteed to find the global minimum of the error function.

Big data driven transportation computational graph to implement back propagation algorithm

Solution procedure to implement back propagation

Dynamic programming

Bound condition: interface to data sources

$$\frac{\partial \mathrm{F}}{\partial v} = 1, \forall v \in \boldsymbol{V}_c^B$$

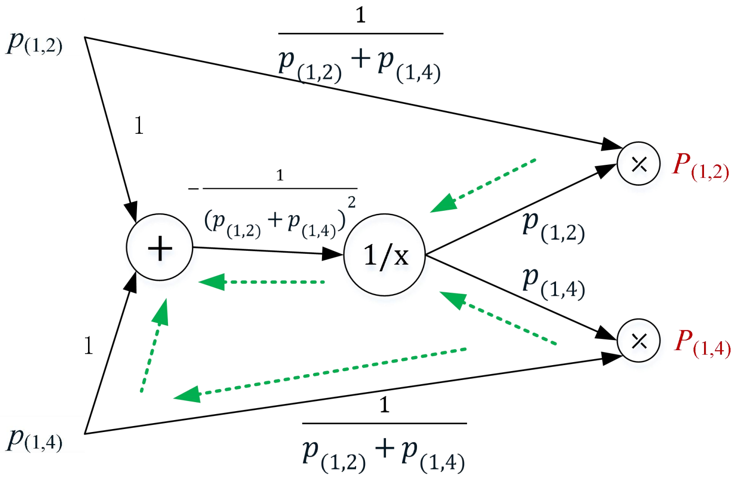
State transition function

$$\frac{\partial \mathrm{F}}{\partial v} = \sum_{v\prime \in \triangle^+(v)} \frac{\partial \mathrm{F}}{\partial v'} \frac{\partial v'}{\partial v}$$

Comparison

$F_j = \min_{i\prime \in \triangle^-(j)}(t_{ij} + F_i)$ shortest path

Let $\boldsymbol{V}_c$ denote the set of vertexes in the computational graph corresponding to MHFN $G(\boldsymbol{V},\boldsymbol{A})$ (or the common term **"state"** in dynamic programming). $\boldsymbol{A}_c$ implies the set of edges in the computational graph corresponding to MHFN Bound conditions is set on the vertexes served as interface between data sources and the computational graph $G_c(\boldsymbol{V}_c,\boldsymbol{A}_c)$. Let $\boldsymbol{V}_c^B \subset \boldsymbol{V}_c$ be the set of bound vertexes.



$$\frac{\partial P_{(1,4)}}{\partial p_{(1,4)}} + \frac{\partial P_{(1,2)}}{\partial p_{(1,4)}} = \frac{1}{p_{(1,4)}+p_{(1,2)}} - \frac{p_{(1,4)}}{\left(p_{(1,4)}+p_{(1,2)}\right)^2} - \frac{p_{(1,2)}}{\left(p_{(1,4)}+p_{(1,2)}\right)^2} = 0$$

■ Backpropagation Algorithm

- Initialize weights (typically random!)
- Keep doing epochs
  - For each example **e** in training set do
    - **forward pass** to compute
      - O = neural-net-output(network,e)
      - miss = (T-O) at each output unit
    - **backward pass** to calculate deltas to weights
    - update all weights
  - end
- until tuning set error stops improving

Chain rule of derivatives

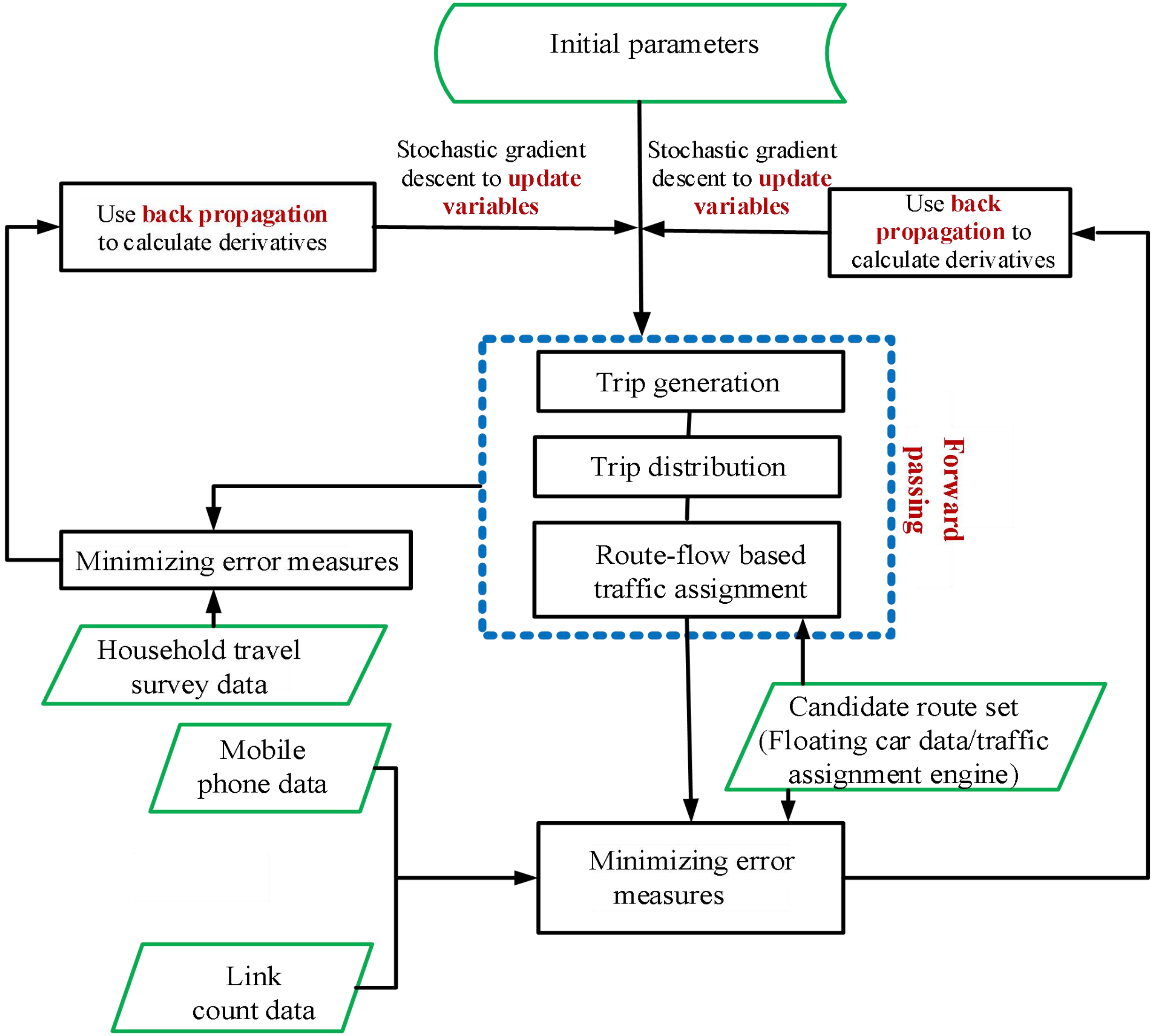
$$\frac{\partial E}{\partial w_{k,j}^{(2.1)}} = \frac{\partial E}{\partial o_k} \frac{\partial o_k}{\partial net_k^{(2)}} \frac{\partial net_k^{(2)}}{\partial w_{k,j}^{(2.1)}}$$

Big data driven transportation computational graph to implement back propagation algorithm

(1) Forward passing
The forward passing process sequentially implements trip generation, trip distribution estimation, and route-based traffic assignment, which is an analogous process of the four-step approach in the field of traffic management.

(2) **Backward propagation**:
The back propagation process inversely implements a feedback control on the forward passing process. Different layers of first-order partial derivatives or "loss errors" are aggregated to calculate marginal gradients.

(3) **Variable values update**:
The estimation variables are updated using stochastic gradient descent using the marginal gradients



Arizona State University

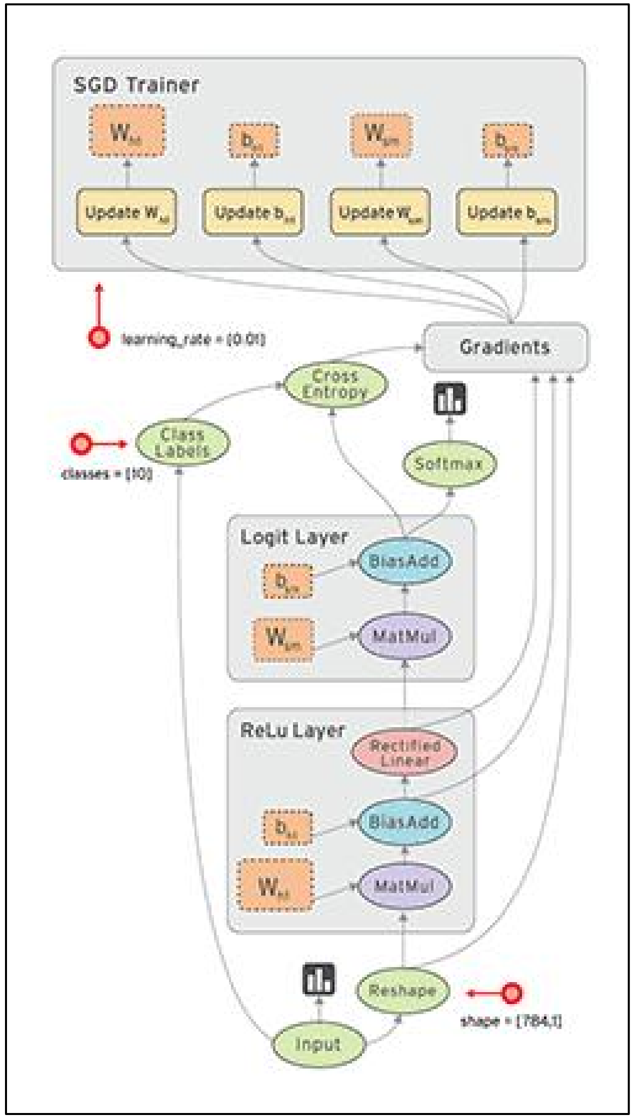
Big data driven transportation computational graph (TCG) to implement back propagation algorithm

- **Automatic differentiation (AD)** is a method to calculate the derivatives of the loss function with respect to the demand variables in a BTCG .
- Derivatives were derived in a **computational graph, using principles of dynamic programming (DP)** (like calculating the shortest path using label correcting algorithm).
- Gradient descent with **Back Propagation (BP) algorithm** guaranteed to find the global minimum of the error function. However, multi-sample-based **stochastic gradient descent** (SGD) can be used to overcome the limit to some extent.
- "AD+DP+BP" can be achieved easily using existing popular data programming tools such as **TensorFlow, Theano** etc.

Arizona State University

An illustrative computational graph from:

https://www.tensorflow.org/guide/graphs?hl=zh-cn



Carefully determine the step sizes in Stochastic Gradient Descent

(1) Vanishing gradient problem in BTCG

$$\frac{\partial F_3(\mathbf{v})}{\partial \boldsymbol{\alpha}} = \frac{\partial F_3(\mathbf{v})}{\partial \mathbf{v}} \times \frac{\partial \mathbf{v}}{\partial \mathbf{f}} \times \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \times \frac{\partial \mathbf{q}}{\partial \boldsymbol{\alpha}}$$

Small values within range [0,1]

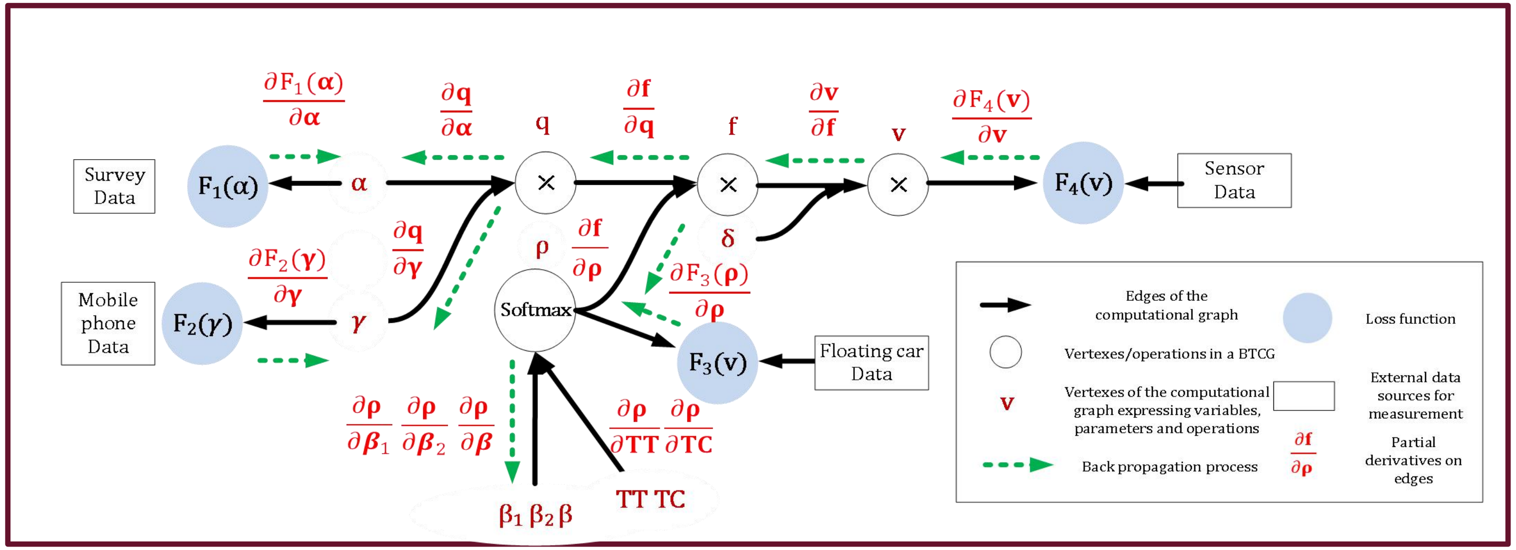
(2) Exploding gradient problem in BTCG

$$\frac{\partial F_4(\mathbf{v})}{\partial \boldsymbol{\gamma}} = \frac{\partial F_3(\mathbf{v})}{\partial \mathbf{v}} \times \frac{\partial \mathbf{v}}{\partial \mathbf{f}} \times \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \times \frac{\partial \mathbf{q}}{\partial \boldsymbol{\gamma}}$$

$$\frac{\partial F_4(\mathbf{v})}{\partial \boldsymbol{\rho}} = \frac{\partial F_3(\mathbf{v})}{\partial \mathbf{v}} \times \frac{\partial \mathbf{v}}{\partial \mathbf{f}} \times \frac{\partial \mathbf{f}}{\partial \boldsymbol{\rho}}$$

Large values equal to OD volume or trip production

Big data driven Transportation Computational Graph (BTCG)



Arizona State University