CIVIL-557
Latent class models - transferability

Evangelos Paschalidis

Transport and Mobility Laboratory (TRANSP-OR)
École Polytechnique Fédérale de Lausanne (EPFL)

EPFL

Previously on Decision Aid Methodologies...

Previous lectures

- Fundamentals of statistical modelling
- Introduction to driving behaviour models
- Car-following models
- Random effects

EPFL

Last week..

- Random effects
- Continuous mixtures
- Error components (random intercept) and random parameters
- Extensions to car-following - random reaction time

Last week's lab..

- Estimation of random effects with simulation
- Estimation with numerical integration
- Estimation with importance sampling

Today's agenda

- Latent class models - discrete mixtures
- Joint model estimation - transferability
- Latent class car-following model (lab)
- Joint model estimation - transferability (lab)

Today's agenda

- Latent class models - discrete mixtures
- Joint model estimation - transferability
- Latent class car-following model (lab)
- Joint model estimation - transferability (lab)

EPFL

Let's revisit the GM model

- Advantage: Acceleration-deceleration asymmetry
- Asymmetry implemented based on the relative speed rule
- We used the normal distribution density function to specify the likelihood function

Let's revisit the GM model - Likelihood function

The density function in acceleration regime is:

$$f(\alpha_n^{acc}(t)|\tau_n) = \frac{1}{\sigma_{\epsilon^{acc}}}\phi\left(\frac{\alpha_n^{acc}(t) - s[X_n^{acc}(t-\tau_n)]f[\Delta V_n(t-\tau_n)]}{\sigma_{\epsilon^{acc}}}\right)$$

The density function in deceleration regime is:

$$f(\alpha_n^{dec}(t)|\tau_n) = \frac{1}{\sigma_{\epsilon^{dec}}}\phi\left(\frac{\alpha_n^{dec}(t) - s[X_n^{dec}(t-\tau_n)]f[\Delta V_n(t-\tau_n)]}{\sigma_{\epsilon^{dec}}}\right)$$

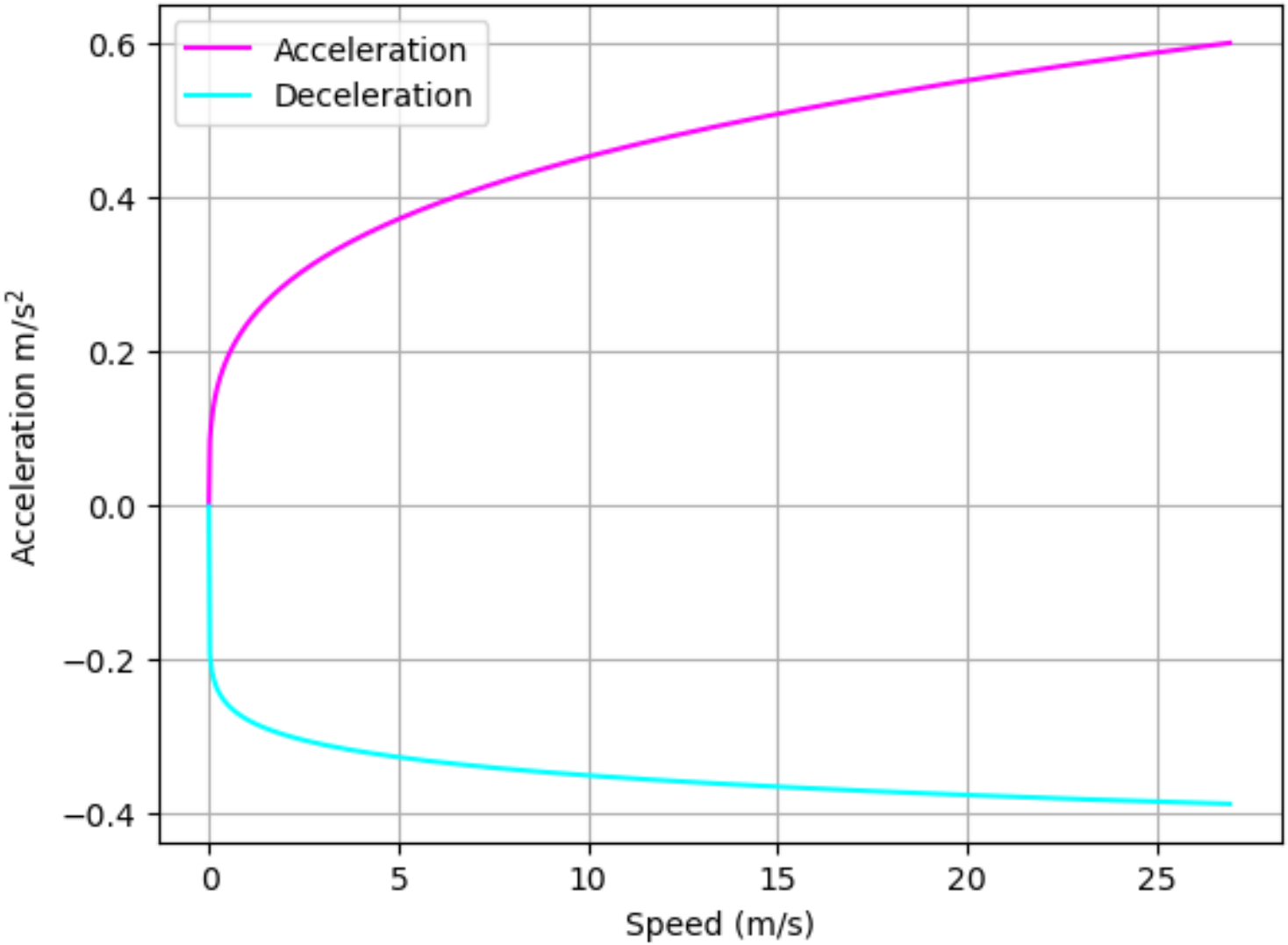
The assumption in this GM model implementation is that a driver will accelerate if relative speed with the lead vehicle is positive (lead vehicle has higher speed compared to following vehicle).

The total acceleration probability of one observation is given by:

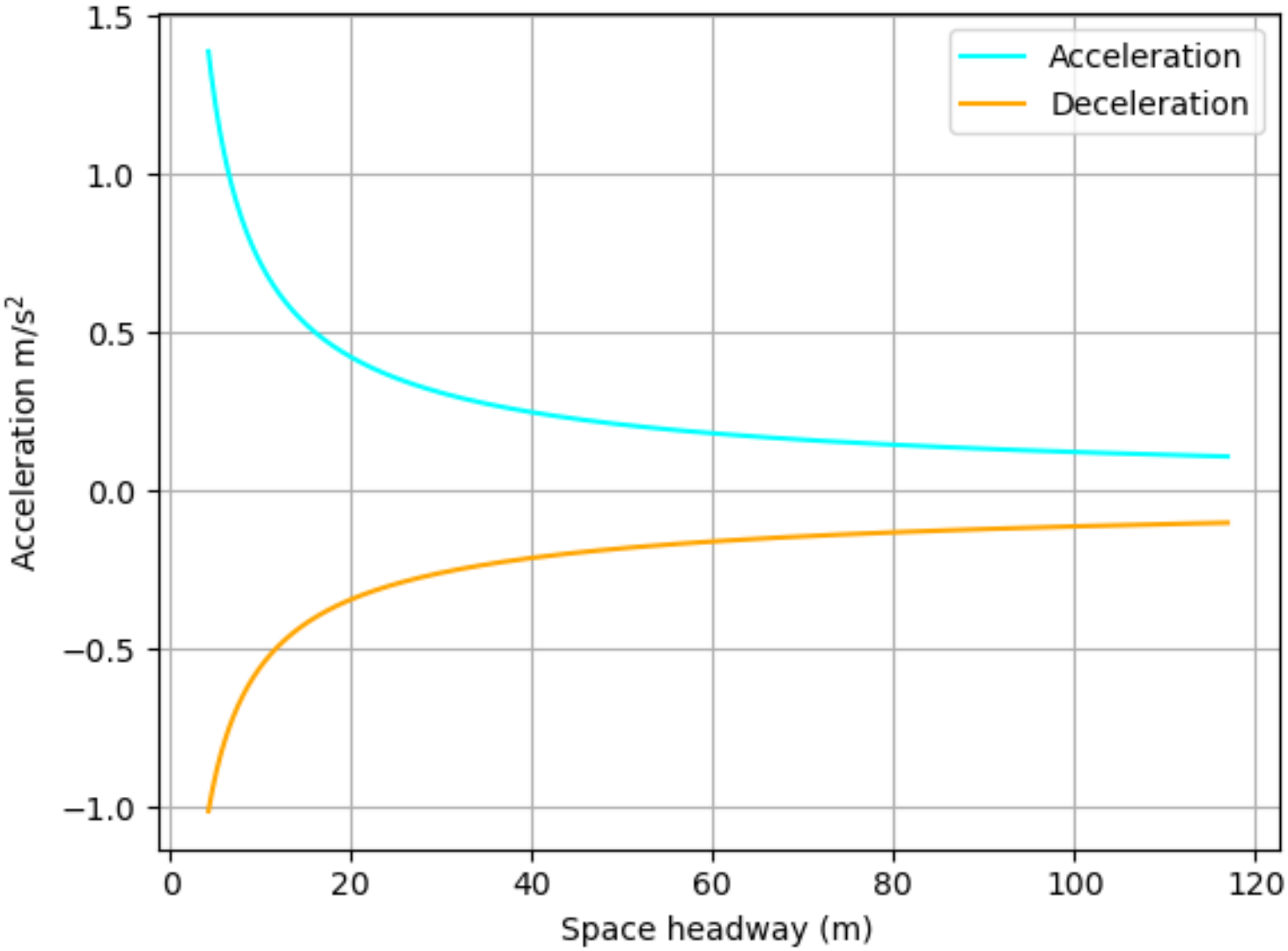
$$f(\alpha_n(t)|\tau_n) = f(\alpha_n^{acc}(t)|\tau_n)^{\Delta V(t-\tau_n)\geq 0} f(\alpha_n^{dec}(t)|\tau_n)^{1-(\Delta V(t-\tau_n)\geq 0)}$$

EPFL

Let's revisit the GM model - Sensitivity analysis



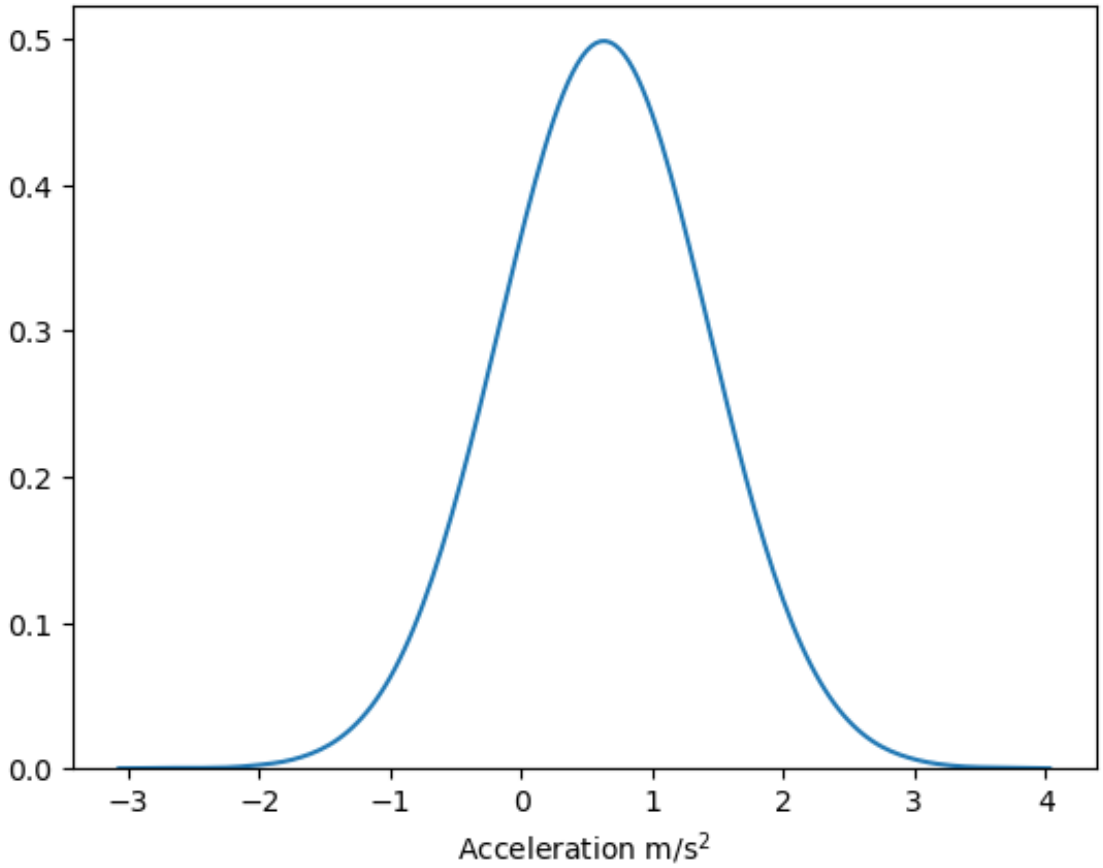
Let's revisit the GM model - Sensitivity analysis



Let's revisit the GM model - Sensitivity analysis

– Let's assume we observe in the data $E(acc) = 0.63 m/s^2$ and $\sigma^{acc} = 0.79 m/s^2$

– If we plot this distribution:



Let's revisit the GM model - Prediction

– The GM model has a multiplicative specification; the whole term takes the sign of the model constant.

$$\alpha_n(t)^g = \alpha^g \frac{V_n(t)^{\beta g}}{\Delta X_n(t)^{\gamma g}} |\Delta V_n(t-\tau_n)|^{\lambda g}$$

– If we apply the asymmetry on an additive specification the model (e.g. Helly's model) may predict negative values in the deceleration state and vice versa.

– Solution: Consider a distribution that corrects the sign issue.

– Suggestions:

- Positive-negative only distributions
- Truncation

EPFL

Let's revisit the GM model - Prediction

– Previously, we assumed that:

$$f(\alpha_n(t)|\tau_n) = f(\alpha_n^{acc}(t)|\tau_n)^{\Delta V(t-\tau_n)\geq 0} f(\alpha_n^{dec}(t)|\tau_n)^{1-(\Delta V(t-\tau_n)\geq 0)}$$

– We could relax this assumption by introducing probabilistic states that capture the intention of a driver to accelerate, decelerate, or do nothing

– The intensity of acceleration (or deceleration) is then decided conditionally on each state

Let's see how random effects can help us here..

Continuous mixtures

Let's assume $w(\theta)$ a positive function so that:

$$\int_\theta w(\theta)\, d\theta = 1$$

then

$$g(\epsilon) = \int_\theta w(\theta) f(\epsilon, \theta)\, d\theta$$

is also a distribution function.

Discrete mixtures

Let's assume w_i and $i = 1, ..., N$ are positive weights so that:

$$\sum_{i=1}^{N} w_i = 1$$

then

$$g(\epsilon; \theta_1, ..., \theta_n) = \sum_{i=1}^{N} w_i f(\epsilon, \theta_i)$$

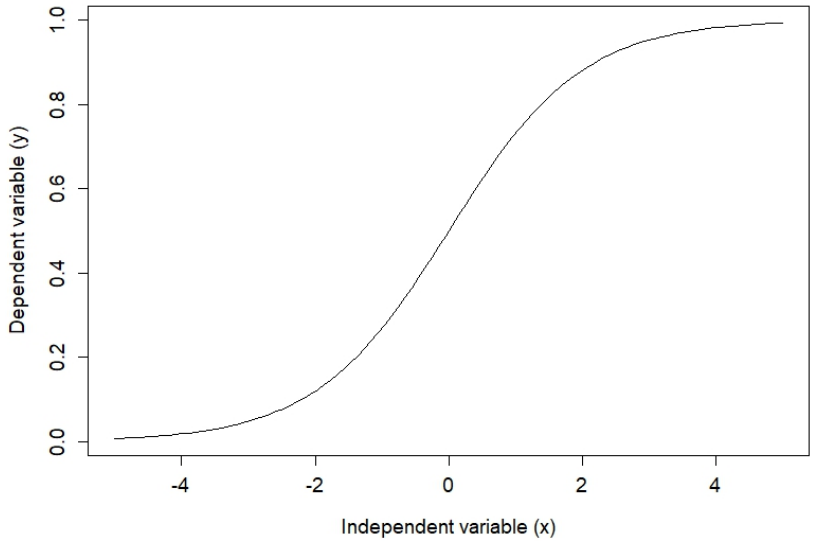
where $f(\epsilon, \theta_i)$ is a parametrised family of distribution functions with θ as parameters to be estimated and ϵ an error term.

We will also see the concept of discrete mixtures in the lane-changing model.

EPFL

Discrete mixtures

Let's assume a binary choice model of choosing alternative i:
$P_i(\beta_1) = \frac{1}{1+e^{(-X\beta_1)}}$

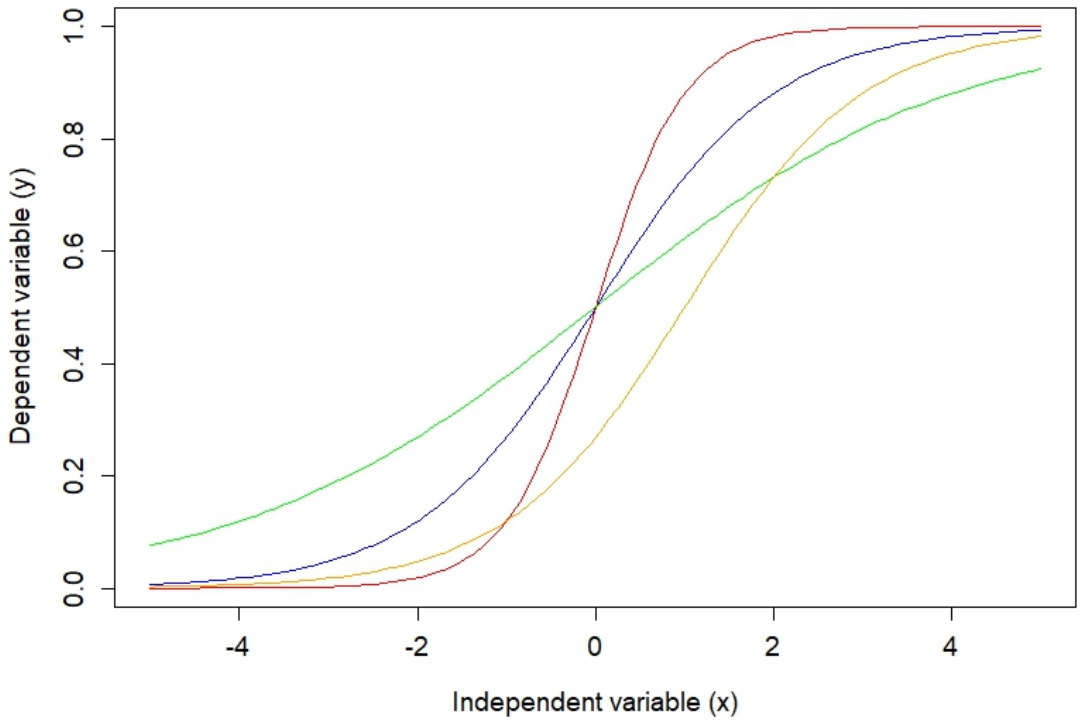


– What if this model does not efficiently capture the behaviour between dependent and independent variable?

Discrete mixtures

$$P_i(\beta_1) = \frac{1}{1+e^{(-X\beta_1)}} \quad P_i(\beta_2) = \frac{1}{1+e^{(-X\beta_2)}}$$

$$P_i(\beta_3) = \frac{1}{1+e^{(-X\beta_3)}} \quad P_i(\beta_4) = \frac{1}{1+e^{(-X\beta_4)}}$$



Discrete mixtures

- In practice, discrete mixtures are another way to add heterogeneity in the models.
- With continuous random effects, we assumed a distribution for the parameter estimates.
- With discrete mixtures we have a discrete number of different models - some of them best representing the individuals of the sample.

EPFL

Discrete mixtures

– Let's assume we want: $0.3P(\beta_1)+0.3P(\beta_2)+0.2P(\beta_3)+0.2P(\beta_4)$

– We need w_i and $i=1,...,N$ are positive weights so that:

$$\sum_{i=1}^{N} w_i = 1$$

– How can we generate numbers that are positive and sum to unity??

Latent classes

– The logit formula can be used to represent the weights w_i (else, class membership probability)

– Let's assume that the sample can be grouped in S groups (classes), each class with a model that best captures observed patterns

– We can define a likelihood function such as

$$L_n(\beta\pi) = \sum_{s=1}^{S} \pi_{ns} \left(\prod_{t=1}^{T_n} P_{ni^*t}(\beta_s) \right)$$

for every individual n with a total of T observations each.

EPFL

Latent classes

– The class membership probability π is estimated as a parameter

– We can use the logit formula to constrain the π values to positive and sum to unity

– We can also independent variables to further explain class membership probabilities

$$\pi_{ns} = \frac{e^{\delta_s + g(\gamma_s, z_n)}}{\sum_{l=1}^{S} e^{\delta_l + g(\gamma_l, z_n)}}$$

where δ_s is a class-specific constant, γ_s is a vector of parameters to be estimated and $g()$ is the utility function of the class allocation

Combining discrete and continuous mixtures

We can include continuous random effects in the models within classes as:

$$L_n(\beta, \pi, \sigma) = \sum_{s=1}^{S} \pi_{ns} \int_{\eta} \prod_{t=1}^{T_n} P_{ni^*t}(\beta_s, \eta) f(\eta|\sigma) d\eta$$

We can include continuous random effects within class allocation probabilities:

$$L_n(\beta, \pi, \sigma) = \int_{\eta} \sum_{s=1}^{S} \pi_{ns}(\eta) \left(\prod_{t=1}^{T_n} P_{ni^*t}(\beta_s) \right) f(\eta|\sigma) d\eta$$

EPFL

Latent class example - 2 classes

Example of a binary choice model with two classes

- We estimate class allocation parameters for class 1

	Class 1		Class 2		Class allocation		
	Estimate	**t-ratio**	**Estimate**	**t-ratio**		**Estimate**	**t-ratio**
asc_{bus}	0.12	5.35	-0.08	-3.40	Income	-0.03	-1.77
asc_{train}	0.22	2.11	-0.15	-7.31	Age	-0.01	-1.97
tt_{car}	-0.03	-1.86	-0.02	-7.93	Car ownership	-0.32	-2.35
tc_{car}	-0.05	-6.37	-0.01	-8.40			
tt_{bus}	-0.06	-2.12	-0.09	-0.23			
tc_{bus}	-0.03	-3.93	-0.02	-3.81			
tt_{train}	-0.05	-1.96	-0.06	-1.74			
tc_{train}	-0.06	-1.07	-0.01	-1.77			

Latent classes summary

- An alternative (discrete) way to incorporate random effects in the models
- More direct interpretation; individuals within each class can be categorised in groups based on the parameter estimates
- A probabilistic method of classification

Any relevance to driving behaviour?

Latent class and car-following

- The asymmetry in the GM model is applied using relative speed as a condition
- This deterministic approach is reasonable but maybe less accurate
- We can use latent classes to relax this assumption

Latent class and car-following

- Let's consider three potential states: (a) acceleration, (b) deceleration, and (c) do nothing.
- We can estimate a model for each state.
- We can treat each state as a latent class and estimate class allocation probabilities
- Acceleration and deceleration classes can be treated as mutually exclusive but we can consider the do nothing state as correlated to both

Latent class and car-following

- In the GM model we considered acceleration-deceleration asymmetry two density functions and relative speed as a condition for acceleration or deceleration.
- Other car-following model specifications do not consider this asymmetry at all.
- We can implement the latent class specification to any car-following model that we can estimate with maximum likelihood.

EPFL

Latent class and car-following

Suggestion for the likelihood function:

- For the acceleration component, we must use a density function that will model just the positive portion of the observation
- Log-normal density function is an option for a positive-only values
- Log-normal distribution has very long tails and can be challenging to estimate
- Alternative: truncated normal considering only the positive part of the distribution

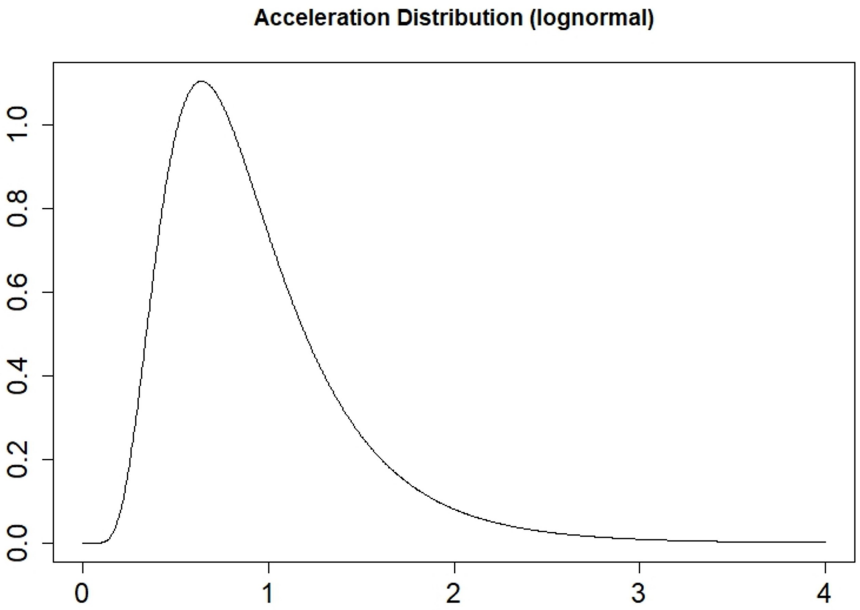
Latent class and car-following

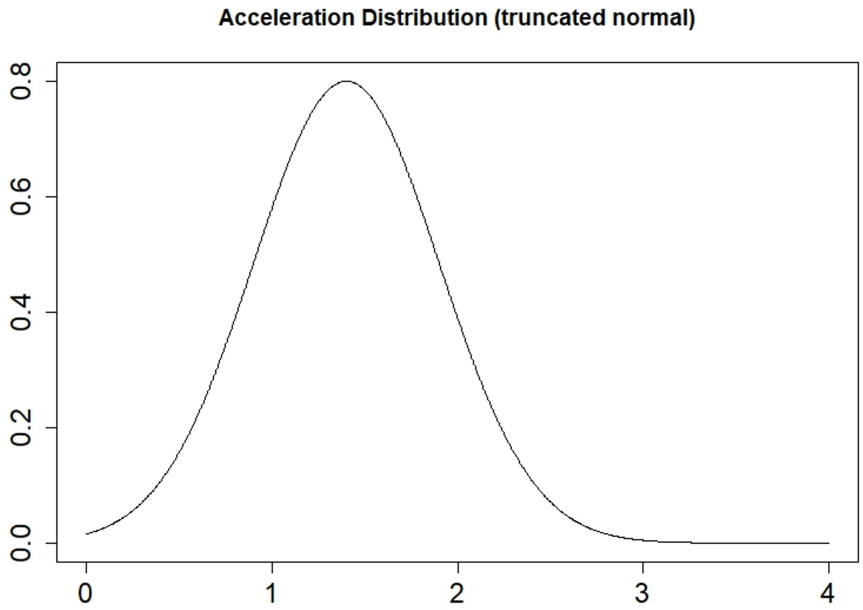
Suggestion for the likelihood function:

- For the deceleration component, the same approach applies as with acceleration
- We can use the negative part of a truncated normal distribution
- If we decide to use a log-normal distribution, we must model the absolute value of deceleration (log-normal distribution takes only positive values)

Latent class and car-following

Examples of approximating acceleration using different distributions





Latent class and car-following

How can we specify a truncated distribution?

– We have already answered this question in the specification of reaction time distribution.

– A generic formula for the specification of a truncated distribution is:

$$f(x)^* = \frac{f(x)}{F(b)-F(a)}$$

where:

$f()$ is the density function of variable x
$F()$ is the cumulative distribution function of variable x
a, b define the minimum and maximum values of truncation

What are the truncation bounds for acceleration and deceleration?

EPFL

Latent class and car-following

– Truncated normal distribution for acceleration:

$$\phi(x^{acc})^* = \frac{\frac{1}{\sigma^{acc}}\phi\left(\frac{x^{acc}-\mu^{acc}}{\sigma^{acc}}\right)}{1-\Phi\left(\frac{-\mu^{acc}}{\sigma^{acc}}\right)}$$

– Truncated normal distribution for deceleration:

$$\phi(x^{dec})^* = \frac{\frac{1}{\sigma^{dec}}\phi\left(\frac{x^{dec}-\mu^{dec}}{\sigma^{dec}}\right)}{\Phi\left(\frac{-\mu^{dec}}{\sigma^{dec}}\right)}$$

where: μ^{acc} and μ^{dec} are replaced by the specification of the car-following model (e.g. *sensitivity x stimulus*)

EPFL

Latent class and car-following

– No truncation is required for the do-nothing state

– The density function of normal distribution can be used:

$$\phi(x^{do-nothing})^* = \frac{1}{\sigma^{dn}}\phi\left(\frac{x^{dn}-\mu^{dn}}{\sigma^{dn}}\right)$$

– For the *do-nothing* state, we do not need to estimate any specific car-following model

– For simplicity, we can estimate a generic normal distribution i.e. estimate two parameters only (mean and standard deviation)

A short note on the specification of the normal density function

```
x = 0.24
mu = 0.17
sigma = 0.33

# PDF specification 1
pdf1 = norm.pdf(x,mu,sigma)

# PDF specification 2
pdf2 = (1/sigma)*norm.pdf((x-mu)/sigma)

print("pdf1 = {}".format(pdf1))
print("pdf2 = {}".format(pdf2))
```

```
pdf1 = 1.1820218293978386
pdf2 = 1.1820218293978386
```

EPFL

Latent class and car-following

Summary (simple implementation):

- We have defined three different car-following states: acceleration, deceleration, and do-nothing
- We have defined the model specification for each state (density function specification)
- We must assign a weight i.e. the probability of being in one of the three states at time t

How can we determine these weights?

EPFL

Latent class and car-following

We can determine the probability of each state using the logit model

- A driver "chooses" whether to accelerate, decelerate or do nothing (discrete component)
- A driver decides "how much" to accelerate, decelerate or do nothing (continuous component) conditionally on the discrete component

EPFL

Latent class and car-following

– The utility of an individual n for accelerating or decelerating at time t can be defined as:

$$U_{nt}^{g} = V_{nt}^{g} + \epsilon_{nt}^{g} = \beta_0^{g} + \beta_k^{g} X_{kt} + \epsilon_{nt}$$

which is a function of k independent variables and $g = (acc, dec)$.

For the do-nothing state we consider $U_{nt}^{dn} = 0$, which serves as reference for the interpretation of parameters.

Latent class and car-following

– The probability of each state for an individual n at time t [$g_n t = (acc, dec, dn)$] is:

$$P_{nt}^{g} = \frac{e^{V_{nt}^{g}}}{\sum_{G} e^{V_{nt}^{G}}}$$

Latent class and car-following

– The total probability of an observation for an individual n at time t is:

$$P_{nt} = P_{nt}^{acc}\phi(x_{nt}^{acc})^* + P_{nt}^{dec}\phi(x_{nt}^{dec})^* + P_{nt}^{dn}\phi(x_{nt}^{dn})^*$$

– However, we need to assign probabilities to the acceleration-deceleration observations in the data. So we can do instead:

$$\begin{aligned} P_{nt} = &[P_{nt}^{acc}\phi(x_{nt}^{acc})^* + P_{nt}^{dn}\phi(x_{nt}^{dn})^*](Acceleration \geq 0) \\ &+ [P_{nt}^{dec}\phi(x_{nt}^{dec})^* + P_{nt}^{dn}\phi(x_{nt}^{dn})^*](Acceleration < 0) \end{aligned} \quad (1)$$

– The specification can be further improved (i.e. the project assignment)

Latent class and car-following

– In this model specification we can add random effects in the form of:

- Reaction time
- Random parameters and/or error components in the car-following models
- Random parameters and/or error components in the utility functions

Today's agenda

- Latent class models - discrete mixtures
- Joint model estimation - transferability
- Latent class car-following model (lab)
- Joint model estimation - transferability (lab)

Model transferability

What is model transferability?

The transfer of a model estimated in one context to a different one.

Motivation:

- To reduce the efforts in the model development (using the same structure of the model previously identified)
- To reduce or eliminate the need for a large data collection in the application context.

Transferability evaluation - Parameter equivalence

t-test of individual parameter equivalence (Galbraith and Hensher, 1982)

– For each of the model parameters it was calculated the term

$$t_{diff} = \frac{\beta_k - \beta_{k*}}{\sqrt{\sigma_k^2 + \sigma_{k*}^2}} \tag{2}$$

where:

β_k: parameter estimates of the estimation (transferred) context model

β_{k*}: parameter estimates of the application context model

σ_k and σ_{k*}, standard errors

If $|t_{diff}| > 1.96$ then significant difference

Transferability evaluation - Transferability Test Statistic (TTS)

Transferability test statistic: assesses whether the null hypothesis of statistical equivalence between the transferred and the application context model is rejected or not.

$$TTS = -2[LL_{k*}(\beta_k) - LL_k(\beta_k)]$$

- LL_{k*}: log-likelihood on the application context data using the transferred context parameters.
- LL_k: log-likelihood on the application context data using application context parameters

The test is assessed like the LR test with degrees of freedom equal to the number of parameters of the application context model.

Improving transferability - Parameter updating

Bayesian updating:

$$\beta_{upt} = \left(\frac{\beta_k}{\sigma_k^2} + \frac{\beta_{k*}}{\sigma_{k*}^2}\right)\left(\frac{1}{\sigma_k^2} + \frac{1}{\sigma_{k*}^2}\right)^{-1}$$

Combined transfer estimation:

$$\beta_{upt} = \left(\frac{\beta_k}{\sigma_k^2 + \alpha\alpha'} + \frac{\beta_{k*}}{\sigma_{k*}^2}\right)\left(\frac{1}{\sigma_k^2 + \alpha\alpha'} + \frac{1}{\sigma_{k*}^2}\right)^{-1}$$

where: $\alpha = \beta_k - \beta_{k*}$ and $\alpha' = \beta_{k*} - \beta_k$

Joint model estimation

- Motivation: The joint estimation of models using various data sources.
- The datasets can have different variables e.g. one of the model variables may be unavailable in one of the data sets.
- The overall process is:
  - For logit model we estimate different constants and scale parameters for each data set. We keep the same parameters for the independent variables.
  - For continuous models we estimate different constants and standard deviation parameters.

EPFL

Joint model estimation - logit example

Let us assume a binary choice model and we use two data sets (a and b) to estimate the model. The utilities for this model are:

$$V_{0a} = ASC_{0a} + \beta_{01} X_{01a} + \beta_{02} X_{02a}$$
$$V_{1a} = \beta_{11} X_{11a} + \beta_{12} X_{12a}$$
$$V_{0b} = \mu_b (ASC_{0b} + \beta_{01} X_{01b} + \beta_{02} X_{02b} + \beta_{03} X_{03b})$$
$$V_{1b} = \mu_b (\beta_{11} X_{11b} + \beta_{12} X_{12b})$$

EPFL

Joint model estimation - continuous model example

For continuous model and data sets a and be we have:

$$\widehat{Y_a} = \beta_{0a} + \beta_1 X_{1a} + \beta_2 X_{2a}$$
$$\widehat{Y_b} = \beta_{0b} + \beta_1 X_{1b} + \beta_2 X_{2b}$$

and

$$f(Y_a) = \frac{1}{\sigma_a}\phi\left(\frac{Y_a - \widehat{Y_a}}{\sigma_a}\right)$$
$$f(Y_b) = \frac{1}{\sigma_b}\phi\left(\frac{Y_b - \widehat{Y_b}}{\sigma_b}\right)$$

$$f(Y) = (Y_a)^{(data==a)}(Y_b)^{(data==b)}$$

EPFL

Joint model estimation - model evaluation

How can we evaluate the model that combines the two datasets? (Example on the linear model)

1. We estimate the joint model as described before (k parameters in total).
2. We estimate the model on Y_a only (n parameters in total).
3. We estimate the model on Y_b only (n parameters in total).
4. we compute the combined LL from steps 2 and 3 $(LL_{Y_a} + LL_{Y_b})$
5. we take the LL value from step 1 as $LL_{Y_{ab}}$
6. We calculate the LR test as $-2[(LL_{Y_a} + LL_{Y_b}) - LL_{Y_{ab}}]$ and evaluate for degrees of freedom $2n - k$.

EPFL

Today's agenda

- Latent class models - discrete mixtures
- Joint model estimation - transferability
- Latent class car-following model (lab)
- Joint model estimation - transferability (lab)

Questions??

A comment on implementing integration

– The typical Monte Carlo simulation implementation is:

$$\int_a^b g(x)f(x)\,dx = \frac{1}{R}\sum_{r=1}^{R} g(x_r)$$

– An implementation of numerical integration with Gauss-Legendre quadrature is:

$$\int_a^b g(x^*)\,dx \approx \frac{b-a}{2}\sum_{i=1}^{N} w_i g\left(\frac{b-a}{2}x_i + \frac{b+a}{2}\right)$$

where x_i takes values between -1 and 1

EPFL

A comment on implementing integration

– It is desirable to convert the numerical integration approach to a specification similar to Monte Carlo so we do not have to change our code. Let's first set:

$\frac{b-a}{2} = C$
$\frac{b-a}{2} x_i + \frac{b+a}{2} = x^*$

– Now it is:

$$\int_a^b g(x^*)\,dx \approx \frac{b-a}{2} \sum_{i=1}^{N} w_i g\left(\frac{b-a}{2} x_i + \frac{b+a}{2}\right) =$$

$$C \sum_{i=1}^{N} w_i g(x^*) = \frac{N}{N} \sum_{i=1}^{N} C w_i g(x^*) = \frac{1}{N} \sum_{i=1}^{N} NC w_i g(x^*)$$

EPFL

Integration with importance sampling

- The purpose of importance sampling is to assist simulated estimation when it is difficult to draw from our target distribution.
- The idea is that instead of a target distribution $f()$ that is difficult to draw from, we draw from a proposal distribution $g()$
- We draw R draws from $g()$ and follow the same process as Monte Carlo simulation by weighting our likelihood function as $f(x^r)/g(x^r)$

Requirements:

- The $f()$ and $g()$ distributions have the same support
- The ratio $f()/g()$ is always finite for every draw x^r

EPFL