Harnessing Variational Auto-Encoders for Fitting Deep Mixed-Effects Models

Dimitris Rizopoulos, Nina van Gerwen, Sten Willemsen

Department of Biostatistics, Erasmus Medical Center Rotterdam

 d.rizopoulos@erasmusmc.nl

1 Background & Motivation

High-dimensional Intensive Longitudinal Data

High-dimensional: many longitudinal outcomes

Intensive: many longitudinal measurements per sample unit

- Examples:
  - ▷ electronic health record data
  - ▷ intensive care monitor data

1 Background & Motivation (cont'd)

GrowIt! App Data: Daily mood changes in young adults

- 1174 participants
  - 123K planned measurements
  - *75K measurements missing*

- Outcomes
  - negative affect (bounded continuous $\in [1, 7]$)
  - loneliness and boredom (dichotomous)

1 Background & Motivation (cont'd)

Objective: Dynamic predictions to guide future interventions

1 Background & Motivation (cont'd)

Common methods

- ▷ Multilevel/Random-Effects Models
  - ↑ interpretable
  - ↑ valid under MAR
  - ↓ scale bad & parametric
- ▷ Machine Learning (e.g., recurrent neural networks)
  - ↑ scale well
  - ↑ semi-parametric
  - ↓ valid under MCAR

1 Background & Motivation (cont'd)

Via a novel combination of machine learning and statistical ideas,

we wish to retain the advantageous features of random effects models, and

overcome the aforementioned difficulties

Deep Generalized Mixed Effects Models

Random Effects Models

- *Conditional Independence Assumptions:* Interrelationships
  - ▷ between longitudinal outcomes
  - ▷ within a longitudinal outcome

  explained via *latent variables*

2 Random Effects Models (cont'd)

General definition

$$\begin{cases} [\boldsymbol{Y}_i \mid \boldsymbol{b}_i] \sim \mathcal{F}_\sigma, \\ g\big\{E(\boldsymbol{Y}_i \mid \boldsymbol{b}_i)\big\} = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i, \\ \\ \boldsymbol{b}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{D}) \end{cases}$$

where

- ▷ $\boldsymbol{Y}_i$ response individual i
- ▷ $g(\cdot)$ link function
- ▷ $\mathcal{F}_\sigma$ conditional distribution

3 Deep Generalized Mixed Models

Definition

$$
\begin{cases}
\boldsymbol{b}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\
\\
[\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{X}_i, \boldsymbol{t}_i; \boldsymbol{\theta}] \sim \mathcal{F}\big(g\{\boldsymbol{\mu}_1(\boldsymbol{X}_i, \boldsymbol{T}_i; \boldsymbol{\beta}) + \boldsymbol{\mu}_2(\boldsymbol{B}_i, \boldsymbol{T}_i; \boldsymbol{\beta})\}, \sigma\big), \\
\\
\boldsymbol{\mu}_1(\boldsymbol{X}_i, \boldsymbol{T}_i; \boldsymbol{\beta}) = \mathsf{dnn}\{(\boldsymbol{X}_i, \boldsymbol{T}_i); \boldsymbol{\beta}_1, \boldsymbol{\lambda}_{\beta 1}\}, \\
\boldsymbol{\mu}_2(\boldsymbol{B}_i, \boldsymbol{T}_i; \boldsymbol{\beta}) = \mathsf{dnn}\{(\boldsymbol{B}_i, \boldsymbol{T}_i); \boldsymbol{\beta}_2, \boldsymbol{\lambda}_{\beta 2}\},
\end{cases}
$$

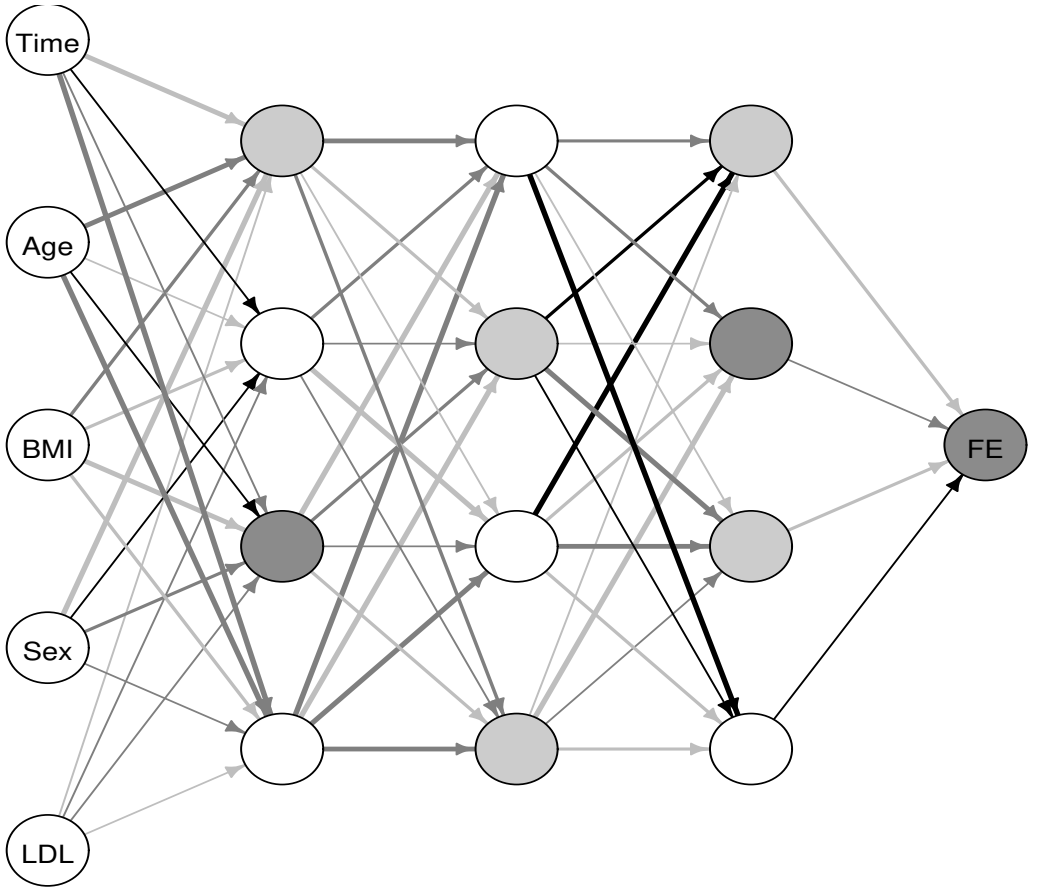
where

- ▷ $\mathsf{dnn}(\boldsymbol{X}; \boldsymbol{\beta}, \boldsymbol{\lambda})$ denotes a dense deep neural network with
- ▷ input variables $\boldsymbol{x}$, parameters $\boldsymbol{\beta}$ and hyper-parameters $\boldsymbol{\lambda}$

3 Deep Generalized Mixed Models (cont'd)

- Notes:
  - ▷ $\boldsymbol{\mu}_1(\boldsymbol{X}_i, \boldsymbol{T}_i; \boldsymbol{\beta})$ *fixed effects neural network*
  - ▷ $\boldsymbol{\mu}_2(\boldsymbol{B}_i, \boldsymbol{T}_i; \boldsymbol{\beta})$ *random effects neural network*
  - ▷ we redefine $\boldsymbol{X}_i$ excluding the time variable $\boldsymbol{t}_i$
  - ▷ matrix $\boldsymbol{T}_i$ contains functions of time

3 Deep Generalized Mixed Models (cont'd)



3 Deep Generalized Mixed Models (cont'd)

Dense Deep Neural Networks

$$\text{dnn}(\boldsymbol{x}; \boldsymbol{\beta}, \boldsymbol{\lambda}) = f_1(\boldsymbol{\beta}_1) \circ \sigma \circ f_2(\boldsymbol{\beta}_2) \circ \ldots \circ \sigma \circ f_L(\boldsymbol{\beta}_L)$$

- ▷ layers $f_1, \ldots, f_L$: tunable affine functions
- ▷ activation function σ: a simple fixed nonlinear function
- ▷ hyper-parameters $\boldsymbol{\lambda}$: number of layers, nodes, ...

Semi-parametric nonlinear interaction models

4 DGMMs Estimation

- Estimation of the model parameters $\boldsymbol{\theta} = (\boldsymbol{\beta}^\top, \sigma)^\top$ requires the marginal likelihood function

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log\{p(\boldsymbol{y}_i; \boldsymbol{\theta})\} = \sum_{i=1}^{n} \log \int p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\, p(\boldsymbol{b}_i)\, \mathrm{d}\boldsymbol{b}_i,$$

  where

  ▷ $p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})$ probability density or probability mass function of $\mathcal{F}(\boldsymbol{\beta}, \sigma)$

  ▷ $p(\boldsymbol{b}_i)$ probability density function of the $\mathcal{N}(\mathbf{0}, \mathbf{I})$

Computational challenge: $\log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\}$ is nonlinear in the random effects $\boldsymbol{b}_i$

4 DGMMs Estimation (cont'd)

We propose an adaption of variational autoencoders to overcome this challenge

- We specify a tractable *variational distribution* for the random effects $q(\boldsymbol{b}_i)$
  - ▷ from which we can easily sample

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \int \frac{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\, p(\boldsymbol{b}_i)}{q(\boldsymbol{b}_i)} q(\boldsymbol{b}_i)\, \mathrm{d}\boldsymbol{b}_i$$

4 DGMMs Estimation (cont'd)

- We use Jensen's inequality to push the logarithm inside the integral

$$
\begin{aligned}
\ell(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \log \int \frac{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\, p(\boldsymbol{b}_i)}{q(\boldsymbol{b}_i)} q(\boldsymbol{b}_i)\, \mathrm{d}\boldsymbol{b}_i \\
&\geq \sum_{i=1}^{n} \int \log\Big\{ \frac{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\, p(\boldsymbol{b}_i)}{q(\boldsymbol{b}_i)} \Big\} q(\boldsymbol{b}_i)\, \mathrm{d}\boldsymbol{b}_i \\
&= \sum_{i=1}^{n} \Big\{ \int \log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\} q(\boldsymbol{b}_i)\, \mathrm{d}\boldsymbol{b}_i - \int \log\Big\{ \frac{q(\boldsymbol{b}_i)}{p(\boldsymbol{b}_i)} \Big\} q(\boldsymbol{b}_i)\, \mathrm{d}\boldsymbol{b}_i \Big\} \\
&= \tilde{\ell}(\boldsymbol{\theta})
\end{aligned}
$$

4 DGMMs Estimation (cont'd)

- When we set $q(\boldsymbol{b}_i) = p(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\theta})$, we obtain that

$$\tilde{\ell}(\boldsymbol{\theta}) \;=\; \ell(\boldsymbol{\theta})$$

Hence, we need a variational density $q(\boldsymbol{b}_i)$ that approximates $p(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\theta})$

4 DGMMs Estimation (cont'd)

Bayesian central limit theorem: As $n_i \to \infty$

$$[\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\theta}] \sim \mathcal{N}\big\{\boldsymbol{\mu}_b(\boldsymbol{y}_i; \boldsymbol{\theta}),\ \boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\theta})\big\}$$

with

$$\boldsymbol{\mu}_b(\boldsymbol{y}_i; \boldsymbol{\theta}) = \underset{b}{\operatorname{argmax}} \Big[\log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}; \boldsymbol{\theta})\} + \log\{p(\boldsymbol{b})\}\Big],$$

and

$$\boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\theta}) = \left\{ -\frac{\partial^2}{\partial \boldsymbol{b}_i \partial \boldsymbol{b}_i^\top} \Big[\log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\} + \log\{p(\boldsymbol{b}_i)\}\Big]\Big|_{\boldsymbol{b}_i = \boldsymbol{\mu}_b} \right\}^{-1}$$

4 DGMMs Estimation (cont'd)

- Both $\boldsymbol{\mu}_b(\boldsymbol{y}_i;\boldsymbol{\theta})$ and $\boldsymbol{\Sigma}_b(\boldsymbol{y}_i;\boldsymbol{\theta})$ can be seeing as nonlinear functions of $\boldsymbol{y}_i$

- We approximate both parameters using deep neural networks again

$$\begin{cases} [\boldsymbol{b}_i \mid \boldsymbol{y}_i;\boldsymbol{\phi}] \sim \mathcal{N}\big\{\boldsymbol{\mu}_b(\boldsymbol{y}_i;\boldsymbol{\phi}),\ \boldsymbol{\Sigma}_b(\boldsymbol{y}_i;\boldsymbol{\phi})\big\}, \\ \\ \big\{\boldsymbol{\mu}_b(\boldsymbol{y}_i;\boldsymbol{\phi}), \boldsymbol{\Sigma}_b(\boldsymbol{y}_i;\boldsymbol{\phi})\big\} = \mathsf{dnn}(\boldsymbol{y}_i;\boldsymbol{\phi},\boldsymbol{\lambda}_\phi) \end{cases}$$

Note: The parameter vector $\boldsymbol{\phi}$ is independent from $\boldsymbol{\theta}$

4 DGMMs Estimation (cont'd)

- dnn$(\boldsymbol{y}_i; \boldsymbol{\phi}, \boldsymbol{\lambda}_\phi)$ a deep neural network that estimates
  - ▷ a $(n \times k)$ matrix of means, and
  - ▷ a $(n \times k')$ matrix of variance components

- $\boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\phi})$ may be taken as a diagonal or a full covariance matrix
  - ▷ k' denotes the number of variance components

4 DGMMs Estimation (cont'd)

Glossary:

- $[\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}]$ is called the *encoder*
  - because it encodes the observed data into a set of latent variables (*dimension reduction*)

- $[\boldsymbol{y}_i \mid \boldsymbol{b}_i, \boldsymbol{X}_i, \boldsymbol{T}_i; \boldsymbol{\theta}]$ is called the *decoder*
  - because it decodes the latent variables to predict the observed data

- *Amortized inference*
  - we approximate $\boldsymbol{\mu}_b(\boldsymbol{y}_i; \boldsymbol{\theta})$ and $\boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\theta})$ using the parameters $\boldsymbol{\phi}$ that are independent of i

4 DGMMs Estimation (cont'd)

- *Loss function:* We minimize

$$-\tilde{\ell}(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\sum_{i=1}^{n} \Bigg\{ \int q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}) \log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\} \, \mathrm{d}\boldsymbol{b}_i + \int q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}) \log\left\{ \frac{q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi})}{p(\boldsymbol{b}_i)} \right\} \mathrm{d}\boldsymbol{b}_i \Bigg\} + \kappa(\boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\phi}^\top \boldsymbol{\phi}),$$

with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$

4 DGMMs Estimation (cont'd)

- *Loss function:* First term using Monte Carlo

$$\int q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}) \log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i; \boldsymbol{\theta})\} \, \mathrm{d}\boldsymbol{b}_i \approx \frac{1}{M} \sum_{m=1}^{M} \log\{p(\boldsymbol{y}_i \mid \boldsymbol{b}_i^{(m)}; \boldsymbol{\theta})\}$$

$$= \frac{1}{M} \sum_{m=1}^{M} \sum_{j=1}^{n_i} \log\{p(y_{ij} \mid \boldsymbol{b}_i^{(m)}; \boldsymbol{\theta})\},$$

with $(\boldsymbol{b}_i^{(1)}, \ldots, \boldsymbol{b}_i^{(M)})$ denoting a sample from $q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi})$

4 DGMMs Estimation (cont'd)

- *Loss function:* Second term

$$\int q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}) \log\left\{\frac{q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi})}{p(\boldsymbol{b}_i)}\right\} \mathrm{d}\boldsymbol{b}_i$$

is the *Kullback-Leibler (KL) divergence* between the variational distribution $q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi})$ and the marginal distribution $p(\boldsymbol{b}_i)$

4 DGMMs Estimation (cont'd)

- *Loss function:* Second term in closed-form
  - ▷ for a diagonal $\boldsymbol{\Sigma}_b(\boldsymbol{y}_i;\boldsymbol{\phi})$ we obtain,

$$\int q(\boldsymbol{b}_i \mid \boldsymbol{y}_i;\boldsymbol{\phi}) \log\Big\{\frac{q(\boldsymbol{b}_i \mid \boldsymbol{y}_i;\boldsymbol{\phi})}{p(\boldsymbol{b}_i)}\Big\}\, \mathrm{d}\boldsymbol{b}_i = \frac{1}{2}\sum_{l=1}^{k} \mu_{il}^2 + \sigma_{il}^2 - 1 - \log(\sigma_{il}^2)$$

  - ▷ for a full covariance matrix

$$\int q(\boldsymbol{b}_i \mid \boldsymbol{y}_i;\boldsymbol{\phi}) \log\Big\{\frac{q(\boldsymbol{b}_i \mid \boldsymbol{y}_i;\boldsymbol{\phi})}{p(\boldsymbol{b}_i)}\Big\}\, \mathrm{d}\boldsymbol{b}_i =$$
$$\frac{1}{2}\Big(\mathsf{trace}\{\boldsymbol{\Sigma}_b(\boldsymbol{y}_i;\boldsymbol{\phi})\} - k + \{\boldsymbol{\mu}_b(\boldsymbol{y}_i;\boldsymbol{\phi})\}^\top\{\boldsymbol{\mu}_b(\boldsymbol{y}_i;\boldsymbol{\phi})\} - \log\big[\mathsf{det}\{\boldsymbol{\Sigma}_b(\boldsymbol{y}_i;\boldsymbol{\phi})\}\big]\Big)$$

4 DGMMs Estimation (cont'd)

- *Loss function:* Last term

$$\kappa(\boldsymbol{\theta}^\top \boldsymbol{\theta} + \boldsymbol{\phi}^\top \boldsymbol{\phi})$$

  - ▷ a ridge penalty to stabilize the parameter estimates and account for overfitting
  - ▷ κ denoting the penalty parameter

4 DGMMs Estimation (cont'd)

Missing Data Challenge

▷ We approximate $q(\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi})$ as

$$\begin{cases} [\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}] \sim \mathcal{N}\big\{\boldsymbol{\mu}_b(\boldsymbol{y}_i; \boldsymbol{\phi}),\ \boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\phi})\big\}, \\ \big\{\boldsymbol{\mu}_b(\boldsymbol{y}_i; \boldsymbol{\phi}), \boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\phi})\big\} = \mathsf{dnn}(\boldsymbol{y}_i; \boldsymbol{\phi}, \boldsymbol{\lambda}_\phi) \end{cases}$$

▷ $\boldsymbol{y}_i$ are 'covariates' in this neural network $\Rightarrow$ *need to have the same length for all subjects*

4 DGMMs Estimation (cont'd)

Because of missing data y_i have unequal lengths

- We decompose $\boldsymbol{y}_i$ to
  - ▷ $\boldsymbol{y}_i^o$ observed measurements
  - ▷ $\boldsymbol{y}_i^m$ unobserved measurements

$$\left\{\boldsymbol{\mu}_b(\boldsymbol{y}_i^o, \boldsymbol{y}_i^m; \boldsymbol{\phi}), \boldsymbol{\Sigma}_b(\boldsymbol{y}_i^o, \boldsymbol{y}_i^m; \boldsymbol{\phi})\right\} = \text{dnn}(\boldsymbol{y}_i^o, \boldsymbol{y}_i^m; \boldsymbol{\phi}, \boldsymbol{\lambda}_\phi)$$

We impute $\boldsymbol{y}_i^m$ under the *Missing At Random* assumption

4 DGMMs Estimation (cont'd)

Utilize *Data Augmentation* ideas

- Target posterior predictive distribution

$$p(\boldsymbol{y}_i^m \mid \boldsymbol{y}_i^o) = \int p(\boldsymbol{y}_i^m \mid \boldsymbol{y}_i^o; \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \boldsymbol{y}_i^o; \boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}$$

4 DGMMs Estimation (cont'd)

- We borrow ideas from the *Stochastic Approximation Expectation Maximization* algorithm

  ▷ Step 0: We set $\boldsymbol{y}_i^{m(0)}$ an initial value for $\boldsymbol{y}_i^m$

For iteration/epoch $it = 1, \ldots, L$*, we repeat the steps*

  ▷ Step 1: Minimization

  estimate $\boldsymbol{\theta}^{(it)}$ and $\boldsymbol{\phi}^{(it)}$ by minimizing $-\tilde{\ell}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{y}_i^{m(it-1)})$

  ▷ Step 2: Imputation

  simulate $\boldsymbol{y}_i^{m(it)}$ from $[\boldsymbol{y}_i^m \mid \boldsymbol{y}_i^o; \boldsymbol{\theta}^{(it)}]$

4 DGMMs Estimation (cont'd)

- The predictive distribution $[\boldsymbol{y}_i^m \mid \boldsymbol{y}_i^o; \boldsymbol{\theta}^{(it)}]$ does not have a closed-form, but is written as

$$p(\boldsymbol{y}_i^m \mid \boldsymbol{y}_i^o; \boldsymbol{\theta}^{(it)}) = \int p(\boldsymbol{y}_i^m \mid \boldsymbol{b}_i; \boldsymbol{\theta}^{(it)})\, p(\boldsymbol{b}_i \mid \boldsymbol{y}_i^o; \boldsymbol{\theta}^{(it)})\, \mathrm{d}\boldsymbol{b}_i$$

- We use a Metropolis-Hastings algorithm

  ▷ simulate $\boldsymbol{b}_i$ from $[\boldsymbol{b}_i \mid \boldsymbol{y}_i^o; \boldsymbol{\theta}^{(it)}]$

  ▷ simulate $\boldsymbol{y}_i^{m(it)}$ from $[\boldsymbol{y}_i^m \mid \boldsymbol{b}_i; \boldsymbol{\theta}^{(it)}]$

4 DGMMs Estimation (cont'd)

Variance Estimation

▷ we have obtained

$$(\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\phi}}) = \underset{\theta,\phi}{\operatorname{argmin}}\{\tilde{\ell}(\boldsymbol{\theta}, \boldsymbol{\phi})\}$$

- To get var$(\widehat{\boldsymbol{\theta}})$ we return to $\ell(\boldsymbol{\theta})$

$$\text{var}(\widehat{\boldsymbol{\theta}}) = \left\{ -\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \Big[\log\{p(\boldsymbol{y}_i; \boldsymbol{\theta})\} \Big] \Big|_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}} \right\}^{-1}$$

▷ Monte Carlo approximation with big M from the encoder to get $\log\{p(\boldsymbol{y}_i; \boldsymbol{\theta})\}$

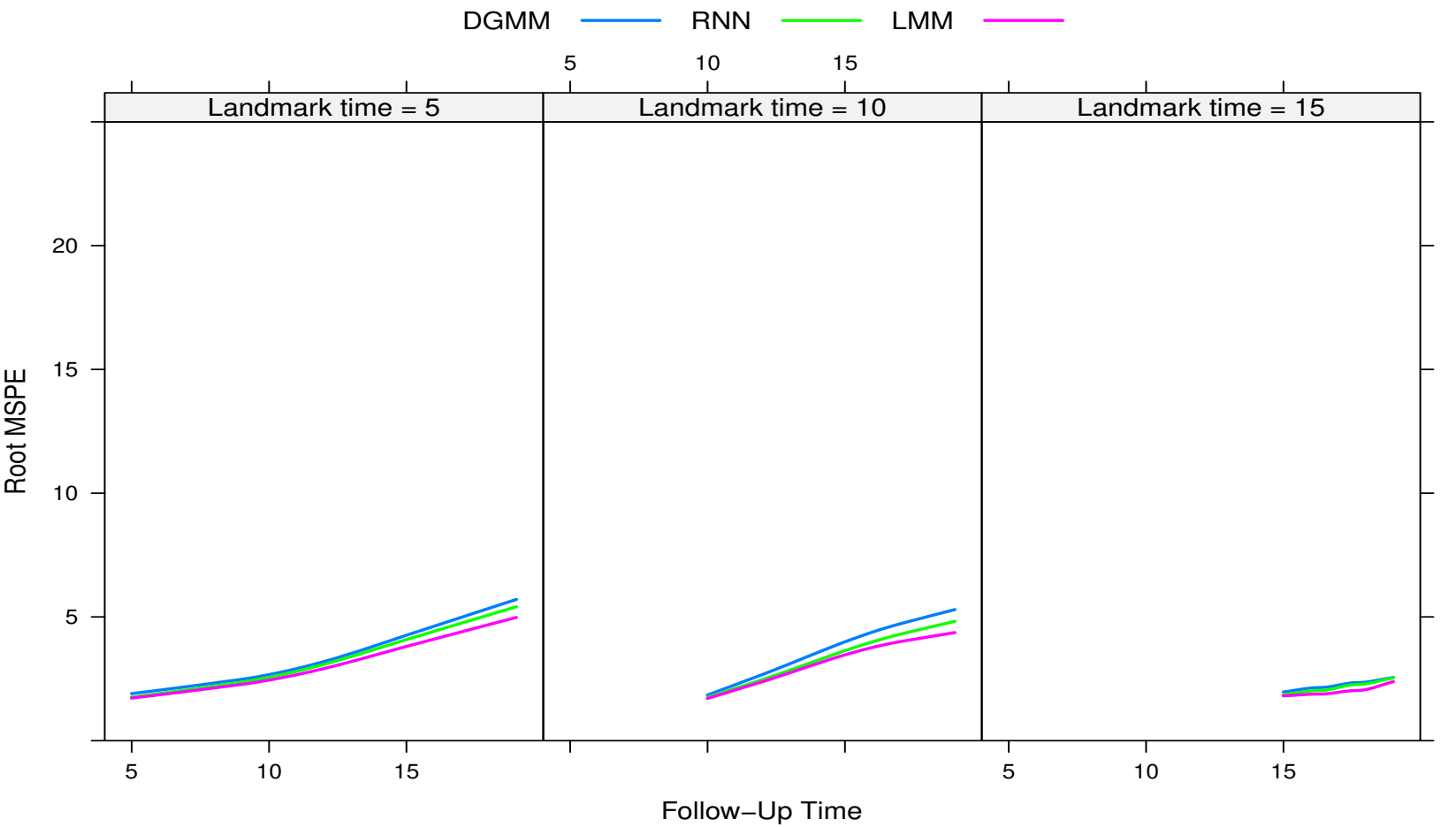
▷ central difference approximation

5 Simulations

DGMM is implemented in Python using Tensorflow/Keras API

- Simulation Study I
  - ▷ one longitudinal outcome, nonlinear subject-specific profiles
  - ▷ $n = 800$, 20 measurements per subject
  - ▷ *no missing data*

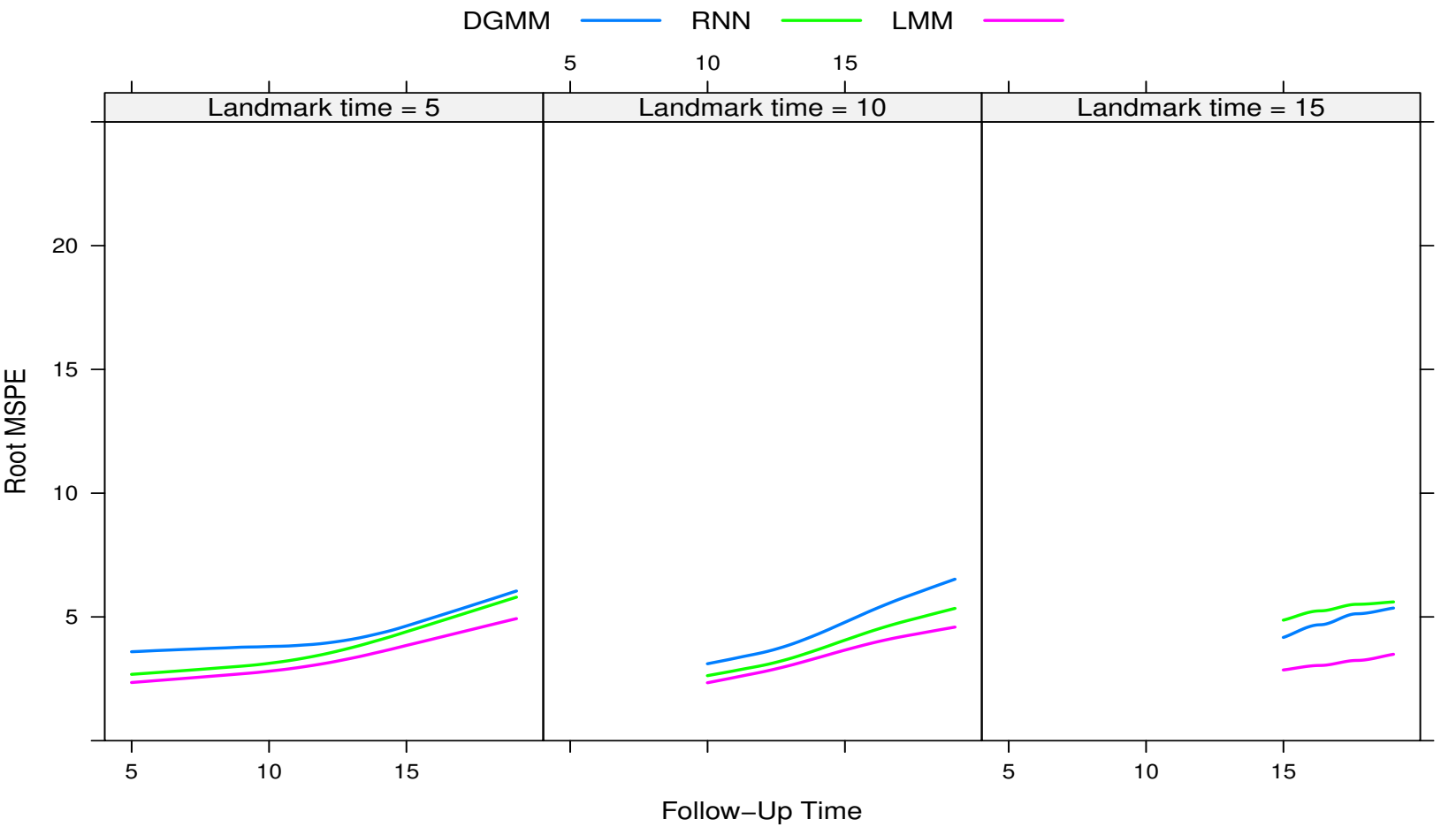
5 Simulations (cont'd)



5 Simulations (cont'd)

- Simulation Study II
  - ▷ one longitudinal outcome, nonlinear subject-specific profiles
  - ▷ $n = 800$, 20 measurements per subject
  - ▷ MAR missing data
  - ▷ 30% dropout

5 Simulations (cont'd)



6 Discussion & Future Plans

- DGGM shows good performance, but not yet as accurate as the linear mixed model
  - ▷ though it scales better for multivariate longitudinal data

- We are looking at ways to improve performance
  - ▷ replace dense with recurrent neural networks

6 Discussion & Future Plans (cont'd)

- Notes / lessons learned:
  - ▷ DNN's can be helped to model nonlinear profiles using functions of time (e.g., polynomials, cosine, etc.) in the specification of $\boldsymbol{T}_i$
  - ▷ a full covariance matrix $\boldsymbol{\Sigma}_b(\boldsymbol{y}_i; \boldsymbol{\phi})$ in the encoder $[\boldsymbol{b}_i \mid \boldsymbol{y}_i; \boldsymbol{\phi}]$ works better than a diagonal matrix
  - ▷ *Posterior collapse*
    - * KL annealing
    - * $n_\phi > n_\theta$

6 Discussion & Future Plans (cont'd)

- Imputation step in DGMM not inherent to the fitting algorithm
  - ▷ *alternative:* pattern mixture imputation scheme
  - ▷ available case missing value restrictions correspond to MAR
  - ▷ decoder and encoder per dropout pattern

- Extend DGMM to fit joint models for longitudinal and event time data
  - ▷ event times are assumed to follow a generic distribution
  - ▷ baseline line hazard is approximated with a dnn
  - ▷ functional relationship between the hazard of an event and covariates and the linear predictor of the longitudinal model approximated with a dnn

Thank for your attention!

https://www.drizopoulos.com/