Structural Equation Models as Computation Graphs

Erik-Jan van Kesteren & Daniel Oberski

Utrecht University, Methodology & Statistics

October 16, 2019
Primer: something else I’m working on
Brain structure covariance matrix (N >600)
EFA with residual structure: a big model
A big model :(

![Graph showing AIC and BIC values for different numbers of EFA factors with convergence and model type indications.](image-url)
A small model?
Some SEM models are overparameterized (e.g., when $P > N$)

- We can't estimate these models with default SEM
- Neural networks can be extremely overparameterized
- Deep learning software (e.g., TensorFlow) can still estimate these
- Can we use deep learning methods for SEM?
Deep learning part 1: adaptive first-order optimizers

Deep learning part 2: computation graphs

The SEM computation graph

Extending SEM

R package showcase
Deep learning part 1: adaptive first-order optimizers
Adaptive first-order optimizers

The loss function is a function of parameters

\[ F(\theta) \]

Outputs a single number, a distance metric (e.g., expected - observed; sum of squared residuals)
In statistics often log-likelihood \( F(\theta) = \ell(\theta|x, y) \)
SEM maximum likelihood loss function

\[ \theta = \{ B_0, \Lambda, \Psi, \Theta \} \]

\[ B = (I - B_0) \]

\[ \Sigma = \Lambda B^{-1} \Psi B^{-T} \Lambda^T + \Theta \]

\[ F_{ML}(\theta) = \log |\Sigma| + tr \left[ S\Sigma^{-1} \right] \]
Optimizers find the parameters $\hat{\theta}$ for which the loss is minimum (maximum)

At the minimum, the gradient of the loss $\nabla F(\theta)$ – the vector of partial derivatives $\left[ \frac{\partial F}{\partial \theta_1}, \frac{\partial F}{\partial \theta_2}, \frac{\partial F}{\partial \theta_3}, \ldots, \frac{\partial F}{\partial \theta_P} \right]$ – is $0$

If there is no closed-form solution for $\nabla F(\theta) = 0$, optimizers may still find the minimum in an iterative way: $\hat{\theta}^{(1)}, \hat{\theta}^{(2)}, \hat{\theta}^{(3)}, \ldots, \hat{\theta}^{(R)}$
Gradient descent

Simplest iterative algorithm: take steps of size $s$ in the direction of the negative gradient (Cauchy, 1847)

$$\hat{\theta}^{(i)} = \hat{\theta}^{(i-1)} - s \cdot \nabla F(\hat{\theta}^{(i-1)})$$
Gradient descent for $F(\theta) = \theta_1^2 + 5\theta_2^2$
Gradient descent can get stuck in local minima and determining the step size is difficult

Deep learning field developed two main improvements to gradient descent

**Momentum** and **adaptive step-size**
Momentum

Instead of using the gradient, use a *moving average* of the history of gradients, for example with a decay of 0.99:

\[
\begin{align*}
v^{(i)} &= 0.99 \cdot v^{(i-1)} + (1 - 0.99) \cdot \nabla F(\hat{\theta}^{(i-1)}) \\
\hat{\theta}^{(i)} &= \hat{\theta}^{(i-1)} - s \cdot v^{(i)}
\end{align*}
\]
Gradient descent with momentum for $F(\theta) = \theta_1^2 + 5\theta_2^2$
https://distill.pub/2017/momentum/
Adaptive step-size

Newer algorithms such as Adam (Kingma & Ba, 2014) also include adaptive step-size

Idea: edit the step size per parameter based on how variable the gradient in that direction is

Less variation in the history of gradients $\rightarrow$ larger steps
Adam for $F(\theta) = \theta_1^2 + 5\theta_2^2$
Adam is one of the most popular optimisation algorithms for deep neural networks

Robust against many kinds of **loss function** abnormalities (e.g., local minima)

Note: for well-behaved* functions, Fisher scoring is still *way* better

*convex, twice continuously differentiable, not too many parameters, etc. (please don’t quote this fuzzy statement)
Deep learning part 2: computation graphs
Computation graphs

Describe operations from parameters to loss function

\[ \theta \rightarrow F(\theta) \]
Least squares regression computation graph

\[ \beta \]

\[ X \]

\[ y \]

\[ F_{LS} \]

nodes:
- multiply
- subtract
- square
- sum

edges:
- \( X \) to \( \beta \)
- \( X \) to \( y \)
- \( y \) to \( F_{LS} \)
Computation graphs

Software can automatically compute $\nabla F(\theta)$ (autograd)
Software implements optimisation algorithms (e.g., Adam)
Computation graphs

Computation graph + software $\rightarrow$ parameter estimation
The SEM computation graph
SEM computation graph

Describe operations from parameters to loss function

$$\theta \rightarrow F(\theta)$$
SEM computation graph

\[ \theta = \{ B_0, \Lambda, \Psi, \Theta \} \]

\[ B = (I - B_0) \]

\[ \Sigma = \Lambda B^{-1} \Psi B^{-T} \Lambda^T + \Theta \]

\[ F_{ML}(\theta) = \log|\Sigma| + tr \left[ S\Sigma^{-1} \right] \]
SEM computation graph

θ
SEM computation graph
SEM computation graph

\[ \theta \rightarrow \Psi \]

\[ \rightarrow B_0 \]

\[ \rightarrow \Lambda \]

\[ \rightarrow \Theta \]
SEM computation graph
SEM computation graph

\[ \begin{align*}
\theta & \rightarrow \psi \\
B_0 & \rightarrow I \\
\Lambda & \rightarrow \psi \\
\Theta & \rightarrow \psi \\
\sum & \rightarrow S
\end{align*} \]
SEM computation graph

\[ \begin{align*}
\theta & \rightarrow \psi \\
& \rightarrow B_0 \\
& \rightarrow \Lambda \\
& \rightarrow \Theta \\
& \rightarrow I \\
\end{align*} \]
SEM computation graph

What can we do now

- Get gradient $\nabla F_{ML}(\theta)$
- Get Hessian $H_{\theta} = H[F_{ML}(\theta)]$
- Get standard errors: $SE_{\theta} \approx \sqrt{\text{diag} \left[H_{\theta}^{-1}\right]}$
- Fit SEM models using smart optimiser (e.g., Adam)
Example
Example
Extending SEM
Extending SEM

Now we can edit the objective:

- Different objective
- Penalise structural paths
- Penalise factor loadings
Least absolute deviation (LAD) is a well-known criterion to fit statistical models, but little is known about LAD estimation in structural equation modeling (SEM). To address this gap, the authors use the LAD criterion in SEM by minimizing the sum of the absolute deviations between the observed and the model-implied covariance matrices. Using Monte Carlo
Least absolute deviation estimation
Least absolute deviation estimation
Least absolute deviation estimation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Absolute value</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambda1</td>
<td>0.0</td>
<td>Lavaan</td>
</tr>
<tr>
<td>Lambda2</td>
<td>0.5</td>
<td>ML_tf</td>
</tr>
<tr>
<td>Lambda3</td>
<td>1.0</td>
<td>LAD_tf</td>
</tr>
<tr>
<td>Lambda4</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

Method: Lavaan, ML_tf, LAD_tf

![Bar chart showing absolute values for different parameters and methods]
Regularized Structural Equation Modeling

Ross Jacobucci, 1 Kevin J. Grimm, 2 and John J. McArdle 1

1 University of Southern California
2 Arizona State University

A new method is proposed that extends the use of regularization in both lasso and ridge regression to structural equation models. The method is termed regularized structural equation modeling (RegSEM). RegSEM penalizes specific parameters in structural equation models, with the goal of creating easier to understand and simpler models. Although regularization has gained wide adoption in regression, very little has transferred to models with latent variables. By adding penalties to specific parameters in a structural equation model, researchers have a high level of flexibility in reducing model complexity, overcoming poor fitting models, and the creation of models that are more likely to generalize to new samples. The proposed method was evaluated through a simulation study, two illustrative examples involving a measurement model, and one empirical example involving the structural part of the model to demonstrate RegSEM’s utility.
Regularized regression

**predictors**

(![](image.png))

10x

unrelated vars
Regularized regression

Method
- glmnet
- regsem
- tensorsem

Regression coefficient

Absolute value

Regression coefficient

Method: glmnet, regsem, tensorsem

49
The problem of penalized maximum likelihood (PML) for an exploratory factor analysis (EFA) model is studied in this paper. An EFA model is typically estimated using maximum likelihood and then the estimated loading matrix is rotated to obtain a sparse representation. Penalized maximum likelihood simulta-
Regularized exploratory factor analysis

\[ \mathbf{F} \]

\[ \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6, \ldots, \mathbf{x}_{10} \]

\( \text{all 0} \)
Regularized exploratory factor analysis
R package showcase

tenorsem
R package showcase

```r
# Install tensorsem
devtools::install_github("vankesteren/tensorsem@computationgraph")
library(tensorsem)
```
# Create a model using lavaan syntax
mod <- "
F1 =~ x1 + x2 + x3
F2 =~ x4 + x5 + x6
F1 ~ F2
"
dat <- lavaan::HolzingerSwineford1939
R package showcase

```r
# create a tf_sem object, similar to lavaan
tensem ← tf_sem(lav_model = mod, data = dat)

# optimise / compute parameter estimates
tensem$train(niter = 10000)

> [loss: 5.71860] [==================================] 65%
```
# plot the loss over iterations
tensem$plot_loss()
R package showcase

tensem$summary()

TensorFlow SEM session

Loss: 5.718596

Sigma:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,] 1.3583698</td>
<td>0.4594367</td>
<td>0.5818981</td>
<td>0.4136796</td>
<td>0.4595856</td>
<td>0.3828115</td>
</tr>
<tr>
<td>[2,] 0.4594367</td>
<td>1.3817839</td>
<td>0.3252543</td>
<td>0.2312279</td>
<td>0.2568872</td>
<td>0.2139740</td>
</tr>
<tr>
<td>[3,] 0.5818981</td>
<td>0.3252543</td>
<td>1.2748649</td>
<td>0.2928609</td>
<td>0.3253597</td>
<td>0.2710081</td>
</tr>
<tr>
<td>[4,] 0.4136796</td>
<td>0.2312279</td>
<td>0.2928609</td>
<td>1.3506645</td>
<td>1.0901316</td>
<td>0.9080243</td>
</tr>
<tr>
<td>[5,] 0.4595856</td>
<td>0.2568872</td>
<td>0.3253597</td>
<td>1.0901316</td>
<td>1.6597858</td>
<td>1.0087876</td>
</tr>
<tr>
<td>[6,] 0.3828115</td>
<td>0.2139740</td>
<td>0.2710081</td>
<td>0.9080243</td>
<td>1.0087876</td>
<td>1.1963584</td>
</tr>
</tbody>
</table>

Psi:

<table>
<thead>
<tr>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,] 0.6475559</td>
<td>0.0000000</td>
</tr>
<tr>
<td>[2,] 0.0000000</td>
<td>0.9812432</td>
</tr>
</tbody>
</table>
R package showcase

Beta:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.4215872</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.0000000</td>
<td></td>
</tr>
</tbody>
</table>

Lambda:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.55895410</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.70794150</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>1.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>1.11096980</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>0.92538160</td>
</tr>
</tbody>
</table>

Theta:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.53641190</td>
<td>0.0000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>1.124980</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.86291510</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.36942130</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.44868250</td>
<td>0.00000000</td>
</tr>
<tr>
<td>0</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.35608940</td>
</tr>
</tbody>
</table>
# add ridge penalty to lambda and refit
tensem$penalties$ridge_lambda ← 0.1
tensem$train(1000)
tensem$plot_loss()
R package showcase

Loss plot

Loss

Iterations

0 2000 4000 6000 8000 10000
6.0 6.5 7.0 7.5 8.0 8.5

62
References


Automatic gradients
Computation Graphs: gradient computation

Autograd: use the chain rule to traverse the graph from objective back to parameters
Deep learning book section 6.5.1, figure 6.10
Gradient

\[ f \]

\[ z \]

\[ y \]

\[ x \]

\[ w \]
Gradient
Gradient
Gradient

\[ \frac{dz}{dwx} \]

\[ \frac{dz}{dy} \]

\[ \frac{dz}{dx} \]

\[ \frac{dz}{dw} \]
Parameter path for LASSO regression. (Early stopping showcase)