Neural Network Methods for Application in Educational Measurement

Geoffrey Converse

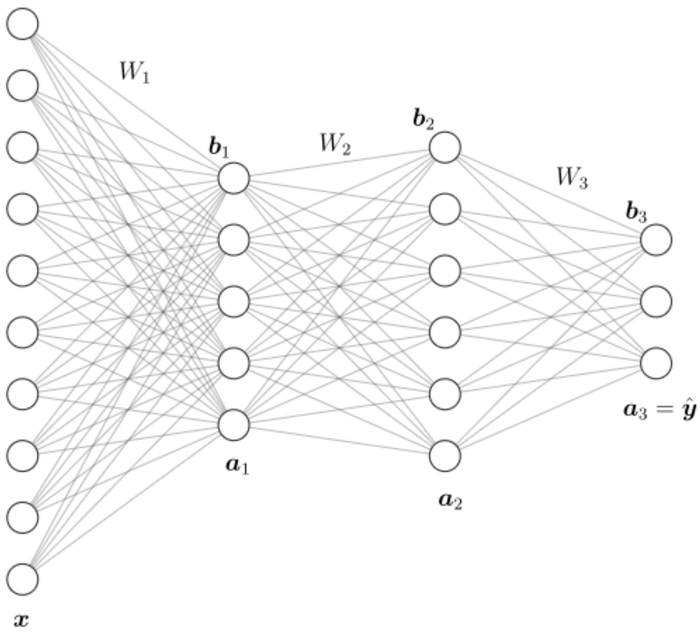
University of Iowa

July 15, 2021

PhD Defense in Applied Mathematical and Computational Sciences

Outline

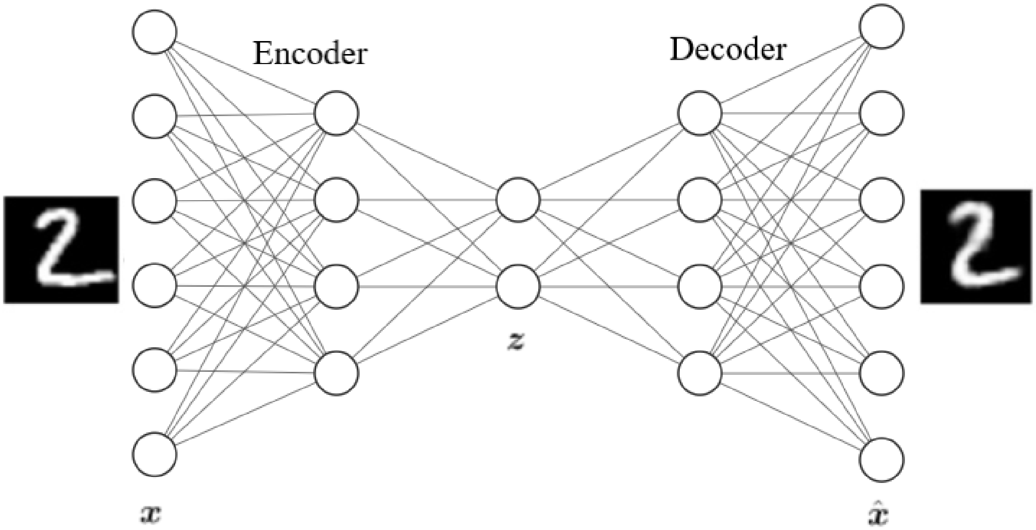
Artificial Neural Networks (ANN)



Input $\boldsymbol{x}$, approximate a true target $\boldsymbol{y}$ via a series of (learned) linear transformations W_l and nonlinear re-scaling $\sigma(\cdot) : \mathbb{R}^d \to (0,1)^d$

$$\hat{\boldsymbol{y}} = \sigma(W_3\sigma(W_2\sigma(W_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3)$$

Autoencoder (AE)



- Encode data into smaller dimension
  - Image compression
  - Non-linear PCA
- Reconstruct original input by minimizing $\mathcal{L} = ||\boldsymbol{x} - \hat{\boldsymbol{x}}||$

Variational Autoencoder (VAE)

- Observed data $\boldsymbol{x}$ is generated by some latent code $\boldsymbol{z}$
- Latent code is assumed to follow a normal distribution $p_z^*(\boldsymbol{z}) = \mathcal{N}(0, I)$
- If $\boldsymbol{z}$ is high dimensional, the posterior is intractable:

$$p_z^*(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_x^*(\boldsymbol{x}|\boldsymbol{z})p_z^*(\boldsymbol{z})}{\int p_x^*(\boldsymbol{x}|\boldsymbol{z})p_z^*(\boldsymbol{z})d\boldsymbol{z}}$$

Variational Autoencoder (VAE)

- Observed data $\boldsymbol{x}$ is generated by some latent code $\boldsymbol{z}$
- Latent code is assumed to follow a normal distribution $p_z^*(\boldsymbol{z}) = \mathcal{N}(0, I)$
- If $\boldsymbol{z}$ is high dimensional, the posterior is intractable:

$$p_z^*(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_x^*(\boldsymbol{x}|\boldsymbol{z})p_z^*(\boldsymbol{z})}{\int p_x^*(\boldsymbol{x}|\boldsymbol{z})p_z^*(\boldsymbol{z})\,d\boldsymbol{z}}$$

- Approximate the true posteriors $p_z^*(\boldsymbol{z}|\boldsymbol{x})$ and $p_x^*(\boldsymbol{x}|\boldsymbol{z})$ with neural networks $q_\alpha(\boldsymbol{z}|\boldsymbol{x})$ and $p_\beta(\boldsymbol{x}|\boldsymbol{z})$

VAE Derivation

$$\begin{aligned}
\log p_x^*(\boldsymbol{x}) &= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log p_x^*(\boldsymbol{x})\, d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p_z^*(\boldsymbol{z}|\boldsymbol{x}) p_x^*(\boldsymbol{x})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \left(\log \frac{q_\alpha(\boldsymbol{z}|\boldsymbol{x})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} + \log \frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{q_\alpha(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z}
\end{aligned}$$

VAE Derivation

$$
\begin{aligned}
\log p_x^*(\boldsymbol{x}) &= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log p_x^*(\boldsymbol{x})\, d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p_z^*(\boldsymbol{z}|\boldsymbol{x}) p_x^*(\boldsymbol{x})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \left(\log \frac{q_\alpha(\boldsymbol{z}|\boldsymbol{x})}{p_z^*(\boldsymbol{z}|\boldsymbol{x})} + \log \frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{q_\alpha(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) || p_z^*(\cdot|\boldsymbol{x}) \right] + \int q_\alpha(\boldsymbol{z}|\boldsymbol{x}) \log \left(\frac{p^*(\boldsymbol{x}, \boldsymbol{z})}{q_\alpha(\boldsymbol{z}|\boldsymbol{x})} \right) d\boldsymbol{z} \\
&= \mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) || p_z^*(\cdot|\boldsymbol{x}) \right] + \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})} \left[- \log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p^*(\boldsymbol{x}, \boldsymbol{z}) \right] \\
&= \mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) || p_z^*(\cdot|\boldsymbol{x}) \right] + \tilde{\mathcal{L}}_*(\alpha; \boldsymbol{x})
\end{aligned}
$$

where $\mathcal{D}_{KL}\left[q||p\right] = \int q(x) \log \left(\frac{q(x)}{p(x)} \right) dx$ measures the information lost from using an approximate distribution p instead of true distribution q

VAE Derivation

- KL-Divergence is non-negative, so we look at the evidence lower bound $\tilde{\mathcal{L}}_*$
- Maximize $\tilde{\mathcal{L}}_* \Rightarrow$ maximize $\log p_x^*(\boldsymbol{x})$

$$\begin{aligned}\log p_x^*(\boldsymbol{x}) \geq \tilde{\mathcal{L}}_*(\alpha; \boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p^*(\boldsymbol{x}, \boldsymbol{z})\right] \\ &= \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_x^*(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z})\right]\end{aligned}$$

VAE Derivation

- KL-Divergence is non-negative, so we look at the evidence lower bound $\tilde{\mathcal{L}}_*$
- Maximize $\tilde{\mathcal{L}}_* \Rightarrow$ maximize $\log p_x^*(\boldsymbol{x})$

$$\begin{aligned}
\log p_x^*(\boldsymbol{x}) \geq \tilde{\mathcal{L}}_*(\alpha; \boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{z}\sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p^*(\boldsymbol{x}, \boldsymbol{z})\right] \\
&= \mathbb{E}_{\boldsymbol{z}\sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_x^*(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z})\right] \\
&\approx \mathbb{E}_{\boldsymbol{z}\sim q_\alpha(\cdot|\boldsymbol{x})}\left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_\beta(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z})\right] \\
&= -\mathcal{D}_{KL}\left[q_\alpha(\cdot|\boldsymbol{x})\middle|\middle|p_z^*(\cdot)\right] + \mathbb{E}_{\boldsymbol{z}\sim q_\alpha(\cdot|\boldsymbol{x})}\left[\log p_\beta(\boldsymbol{x}|\boldsymbol{z})\right] \\
&= \tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x})
\end{aligned}$$

VAE Derivation

- KL-Divergence is non-negative, so we look at the evidence lower bound $\tilde{\mathcal{L}}_*$
- Maximize $\tilde{\mathcal{L}}_* \Rightarrow$ maximize $\log p_x^*(\boldsymbol{x})$

$$
\begin{aligned}
\log p_x^*(\boldsymbol{x}) \geq \tilde{\mathcal{L}}_*(\alpha; \boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})} \left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p^*(\boldsymbol{x}, \boldsymbol{z}) \right] \\
&= \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})} \left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_x^*(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z}) \right] \\
&\approx \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})} \left[-\log q_\alpha(\boldsymbol{z}|\boldsymbol{x}) + \log p_\beta(\boldsymbol{x}|\boldsymbol{z}) + \log p_z^*(\boldsymbol{z}) \right] \\
&= -\mathcal{D}_{KL} \left[q_\alpha(\cdot|\boldsymbol{x}) \middle|\middle| p_z^*(\cdot) \right] + \mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})} \left[\log p_\beta(\boldsymbol{x}|\boldsymbol{z}) \right] \\
&= \tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x})
\end{aligned}
$$

- We've replaced all unknown distributions $p_\cdot^*(\cdot)$ with assumed or approximate distributions
- VAE loss is given as $\mathcal{L}(\alpha, \beta; \boldsymbol{x}) = -\tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x})$ where α and β reference the trainable parameters in the encoder and decoder

VAE Derivation

- Binary data: $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$
  - $p_\beta(\boldsymbol{x}|\boldsymbol{z})$ is a Bernoulli distribution:
  - $\hat{\boldsymbol{x}} = (p_\beta(x_1 = 1|\boldsymbol{z}), \ldots, p_\beta(x_n = 1|\boldsymbol{z}))^\top$

VAE Derivation

- Binary data: $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$
  - $p_\beta(\boldsymbol{x}|\boldsymbol{z})$ is a Bernoulli distribution:
  - $\hat{\boldsymbol{x}} = (p_\beta(x_1 = 1|\boldsymbol{z}), \ldots, p_\beta(x_n = 1|\boldsymbol{z}))^\top$

$$
\begin{aligned}
\mathcal{L}(\alpha, \beta; \boldsymbol{x}) &= -\tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x}) \\
&= -\mathbb{E}_{\boldsymbol{z}\sim q_\alpha(\cdot|\boldsymbol{x})}\left[\log p_\beta(\boldsymbol{x}|\boldsymbol{z})\right] + \mathcal{D}_{KL}\left[q_\alpha(\cdot|\boldsymbol{x})\middle|\middle|p_z^*(\cdot)\right] \\
&= \sum_{i=1}^{n} -x_i \log p_\beta(x_i = 1|\boldsymbol{z}) - (1 - x_i)\log p_\beta(x_i = 0|\boldsymbol{z}) \\
&\quad + \mathcal{D}_{KL}\left[q_\alpha(\cdot|\boldsymbol{x})\middle|\middle|p_z^*(\cdot)\right]
\end{aligned}
$$

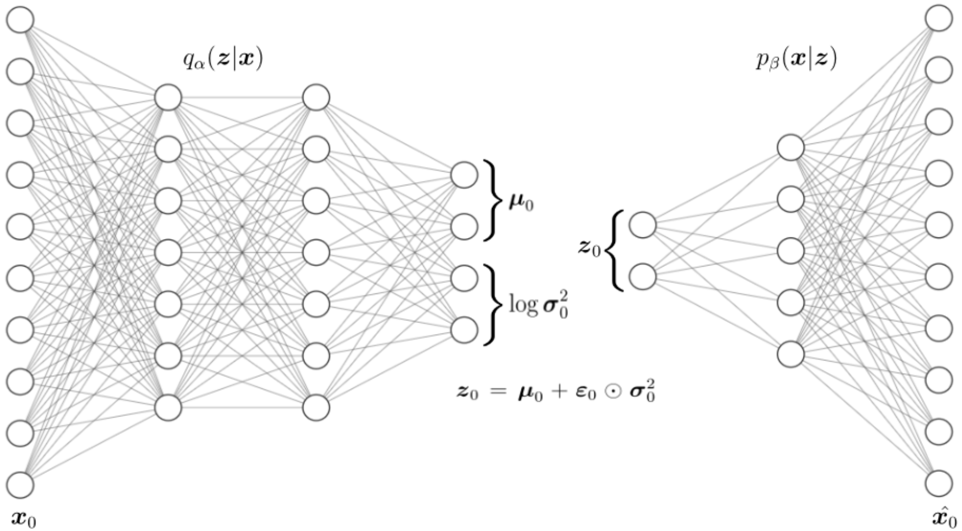
VAE Derivation

- Binary data: $\boldsymbol{x} = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$
  - $p_\beta(\boldsymbol{x}|\boldsymbol{z})$ is a Bernoulli distribution:
  - $\hat{\boldsymbol{x}} = (p_\beta(x_1 = 1|\boldsymbol{z}), \ldots, p_\beta(x_n = 1|\boldsymbol{z}))^\top$

$$\begin{aligned}
\mathcal{L}(\alpha, \beta; \boldsymbol{x}) &= -\tilde{\mathcal{L}}(\alpha, \beta; \boldsymbol{x}) \\
&= -\mathbb{E}_{\boldsymbol{z} \sim q_\alpha(\cdot|\boldsymbol{x})}\left[\log p_\beta(\boldsymbol{x}|\boldsymbol{z})\right] + \mathcal{D}_{KL}\left[q_\alpha(\cdot|\boldsymbol{x}) \middle|\middle| p_z^*(\cdot)\right] \\
&= \sum_{i=1}^{n} -x_i \log p_\beta(x_i = 1|\boldsymbol{z}) - (1 - x_i) \log p_\beta(x_i = 0|\boldsymbol{z}) \\
&\quad + \mathcal{D}_{KL}\left[q_\alpha(\cdot|\boldsymbol{x}) \middle|\middle| p_z^*(\cdot)\right]
\end{aligned}$$

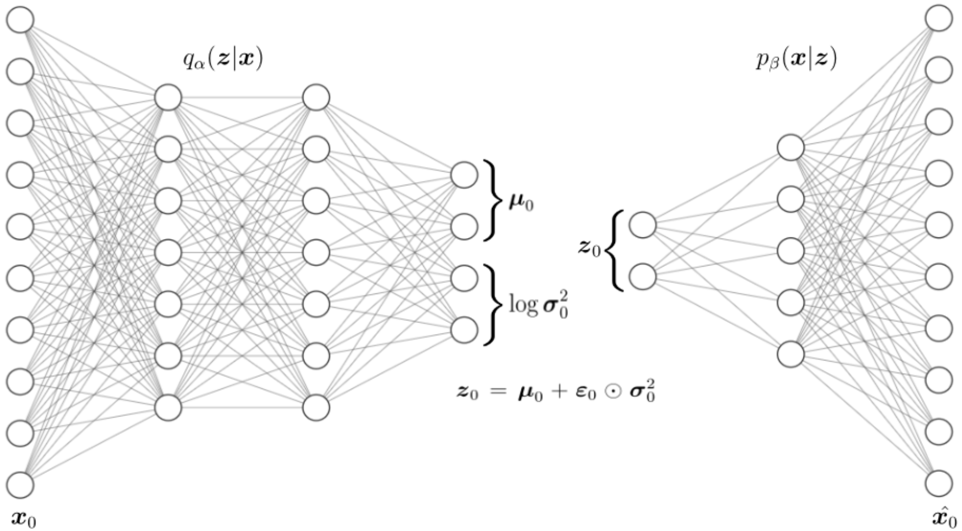
- VAE loss is the binary cross-entropy loss function with a KL-divergence regularizer on the latent code

VAE Architecture



- Fit encoded space to $\boldsymbol{z} \sim \mathcal{N}(0, I)$
- Given input $\boldsymbol{x}_0$, the encoder outputs a distribution $q_\alpha(\boldsymbol{z}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$

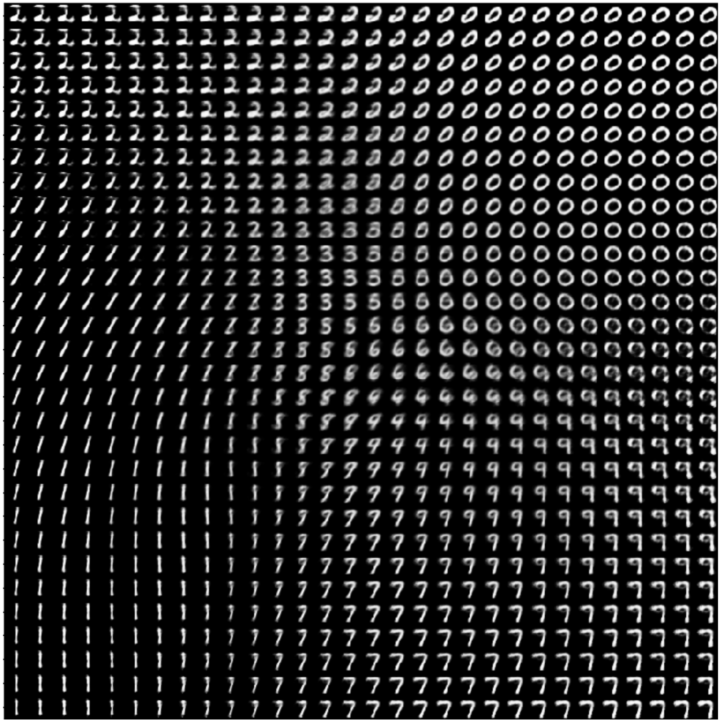
VAE Architecture



- Fit encoded space to $\boldsymbol{z} \sim \mathcal{N}(0, I)$
- Given input $\boldsymbol{x}_0$, the encoder outputs a distribution $q_\alpha(\boldsymbol{z}|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2)$
- Sample $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, I)$, set $\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\varepsilon} \odot \boldsymbol{\sigma}_0$
- Feed $\boldsymbol{z}_0$ through decoder to obtain reconstruction $\hat{\boldsymbol{x}}_0 \sim p_\beta(\cdot|\boldsymbol{z}_0)$

VAE Applications

- VAE are used as a generative model
- Train on a set of images, then generate *new* images which are similar to the training data by sampling form the latent space



Item Response Theory (IRT)

- Goal: Explain relationship between student ability and exam performance
- Each student has a latent "ability" value $\theta \in \mathbb{R}$

Item Response Theory (IRT)

- Goal: Explain relationship between student ability and exam performance
- Each student has a latent "ability" value $\theta \in \mathbb{R}$
  - θ is not directly observable
  - Naive solution: $\theta \approx \dfrac{\text{questions answered correctly}}{\text{total number of questions}}$

Item Response Theory (IRT)

- Goal: Explain relationship between student ability and exam performance
- Each student has a latent "ability" value $\theta \in \mathbb{R}$
  - θ is not directly observable
  - Naive solution: $\theta \approx \dfrac{\text{questions answered correctly}}{\text{total number of questions}}$
- What is the probability that student j answers item i correctly?

$$P(u_{ij} = 1 | \theta_j) = f(\theta_j; \Lambda_i)$$

  - θ_j = latent ability of subject j
  - Λ_i = set of parameters for item i (e.g. difficulty)

Rasch Model

- Define $\delta_i > 0$ as the difficulty of item i, and $\eta_j > 0$ the ability of subject j.
- Rasch: Probability of success depends on ratio $\dfrac{\delta_i}{\eta_j}$

$$P(u_{ij} = 1|\eta_j, \delta_i) = \frac{1}{1 + \delta_i/\eta_j} = \frac{\eta_j}{\eta_j + \delta_i}$$

Rasch Model

- Define $\delta_i > 0$ as the difficulty of item i, and $\eta_j > 0$ the ability of subject j.
- Rasch: Probability of success depends on ratio $\dfrac{\delta_i}{\eta_j}$

$$P(u_{ij} = 1 | \eta_j, \delta_i) = \frac{1}{1 + \delta_i / \eta_j} = \frac{\eta_j}{\eta_j + \delta_i}$$

- Logarithmic transformation: $\theta_j = \log \eta_j$ and $\beta_i = \log \delta_i$
- Rasch Model:

$$P(u_{ij} = 1 | \theta_j, \beta_i) = \frac{1}{1 + e^{\beta_i - \theta_j}}$$

2-Parameter Logistic Model (2PL)

- Probability of a correct response follows the logistic equation:

$$P(u_{ij} = 1 | \theta_j; a_i, b_i) = \frac{1}{1 + e^{-a_i(\theta_j - b_i)}}$$

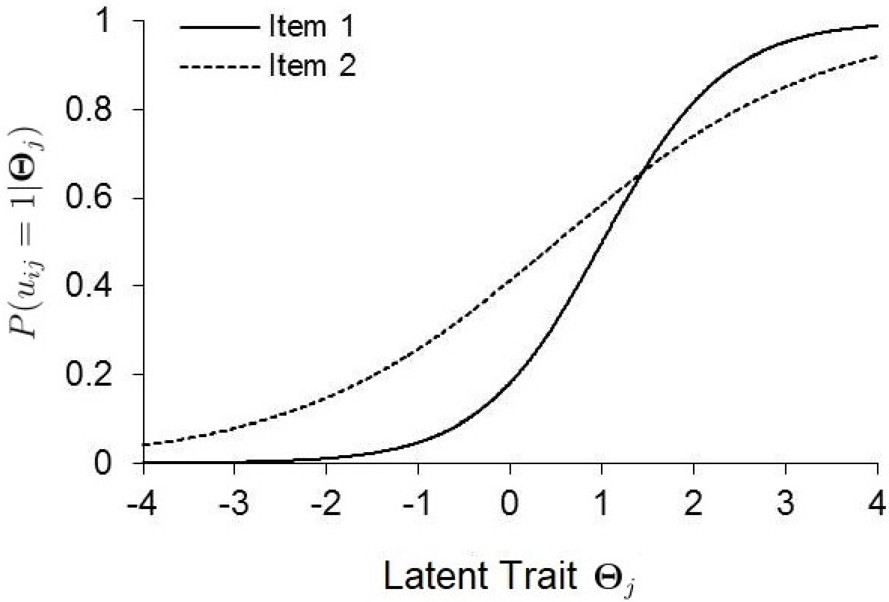
2-Parameter Logistic Model (2PL)

- Probability of a correct response follows the logistic equation:

$$P(u_{ij} = 1 | \theta_j; a_i, b_i) = \frac{1}{1 + e^{-a_i(\theta_j - b_i)}}$$

- a_i = discrimination parameter (slope)
  - Quantifies the capability of item i in differentiating between students with sufficient/insufficient ability
- b_i = difficulty parameter (intercept)

Item Characteristic Curve (ICC)



- Item 1 has higher discrimination than Item 2

Multidimensional IRT

- Now assume that an assessment is testing K skills
  - For example, a math exam can test skills add, subtract, multiply, divide
  - Student j has a vector of skills $\mathbf{\Theta}_j = (\theta_{j1}, .., \theta_{jK})^T$
  - Multiple skills can be assessed by a single item

Multidimensional IRT

- Now assume that an assessment is testing K skills
  - For example, a math exam can test skills add, subtract, multiply, divide
  - Student j has a vector of skills $\boldsymbol{\Theta}_j = (\theta_{j1}, .., \theta_{jK})^T$
  - Multiple skills can be assessed by a single item
- Binary Q-matrix defines relationship between items and skills
  - $Q \in \mathbb{R}^{n \times K}$,

$$q_{ik} = \begin{cases} 1 & \text{if item } i \text{ requires skill } k \\ 0 & \text{otherwise} \end{cases}$$

Multidimensional Logistic 2-Parameter (ML2P) Model

- Probability of correct response given by:[1]

$$P(u_{ij} = 1 | \boldsymbol{\Theta}_j; \boldsymbol{a}_i, b_i) = \frac{1}{1 + \exp\left[-\boldsymbol{a}_i^\top \boldsymbol{\Theta}_j + b_i\right]}$$
$$= \frac{1}{1 + \exp[-\sum_{k=1}^{K} q_{ik} a_{ik} \theta_{jk} + b_i]}$$

[1] da Silva et al., 2018

Multidimensional Logistic 2-Parameter (ML2P) Model

- Probability of correct response given by:[1]

$$P(u_{ij} = 1 | \boldsymbol{\Theta}_j; \boldsymbol{a}_i, b_i) = \frac{1}{1 + \exp\left[-\boldsymbol{a}_i^\top \boldsymbol{\Theta}_j + b_i\right]}$$
$$= \frac{1}{1 + \exp[-\sum_{k=1}^{K} q_{ik} a_{ik} \theta_{jk} + b_i]}$$

  - a_{ik} = discrimination parameter between item i and skill k
  - b_i = difficulty parameter

[1] da Silva et al., 2018

Estimating IRT Parameters

- In application, given only binary matrix of N response sets $U \in \mathbb{R}^{N \times n}$
  - $\boldsymbol{u}_j \in \mathbb{R}^n$ details student j's correct/incorrect responses to n items
- How to obtain the item parameters $\boldsymbol{a}_i$ and b_i and student ability parameters $\boldsymbol{\Theta}_j$?[2]

[2] Baker and Kim, 2004

Estimating IRT Parameters

- In application, given only binary matrix of N response sets $U \in \mathbb{R}^{N \times n}$
  - $\boldsymbol{u}_j \in \mathbb{R}^n$ details student j's correct/incorrect responses to n items
- How to obtain the item parameters $\boldsymbol{a}_i$ and b_i and student ability parameters $\boldsymbol{\Theta}_j$?[2]
- Maximize the log-likelihood of the data

$$\log L = \sum_{j=1}^{N} \sum_{i=1}^{n} u_{ij} \log P(u_{ij} = 1) + (1 - u_{ij}) \log P(u_{ij} = 0)$$

[2] Baker and Kim, 2004

Difficulties of Estimating IRT Parameters

High dimensional IRT is hard

- Joint Maximum Likelihood Estimation (JMLE)[3]
    - Newton-Raphson iterations
        - Large matrix inversions
    - Possibly unbounded estimates
    - Requires anchoring procedure

[3] Chen et al., 2019

[4] Bock and Aitken, 1981

Difficulties of Estimating IRT Parameters

High dimensional IRT is hard

- Joint Maximum Likelihood Estimation (JMLE)[3]
  - Newton-Raphson iterations
    - Large matrix inversions
  - Possibly unbounded estimates
  - Requires anchoring procedure
- Marginal Maximum Likelihood Estimation (MMLE)[4]
  - EM algorithm requires computing a high-dimensional integral:
    - $\boldsymbol{\Theta} \in \mathbb{R}^{20} \implies$ 20-dimensional integral
    - 3 quadrature points per dimension $\implies 3^{20} = 3,486,784,401$
  - Separate estimation of student parameters

---

[3] Chen et al., 2019

[4] Bock and Aitken, 1981

Similarities between IRT and VAE

- IRT and VAE assume normally distributed latent space
  - Observed data is generated from latent code

Similarities between IRT and VAE

- IRT and VAE assume normally distributed latent space
  - Observed data is generated from latent code
- ML2P model and sigmoidal activation function:

$$P(u_{ij} = 1 | \boldsymbol{\Theta}_j) = \frac{1}{1 + \exp[-\sum_{k=1}^{K} a_{ik}\theta_{jk} + b_i]}$$

$$\sigma(z) = \sigma(\boldsymbol{w}^\top \boldsymbol{a} + b) = \frac{1}{1 + \exp[-\sum_k w_k a_k - b]}$$

Similarities between IRT and VAE

- IRT and VAE assume normally distributed latent space
  - Observed data is generated from latent code
- ML2P model and sigmoidal activation function:

$$P(u_{ij} = 1 | \mathbf{\Theta}_j) = \frac{1}{1 + \exp[-\sum_{k=1}^{K} a_{ik}\theta_{jk} + b_i]}$$

$$\sigma(z) = \sigma(\boldsymbol{w}^\top \boldsymbol{a} + b) = \frac{1}{1 + \exp[-\sum_k w_k a_k - b]}$$

- Log-likelihood and binary cross-entropy loss

Model Description[5]

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$

[5] First presented at the International Joint Conference on Neural Networks (IJCNN) 2019

Model Description[5]

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$

[5] First presented at the International Joint Conference on Neural Networks (IJCNN) 2019

Model Description[5]

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$
- Sigmoidal activation function in output layer

[5] First presented at the International Joint Conference on Neural Networks (IJCNN) 2019

Model Description[5]

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$
- Sigmoidal activation function in output layer
- Decoder interpreted as the ML2P model

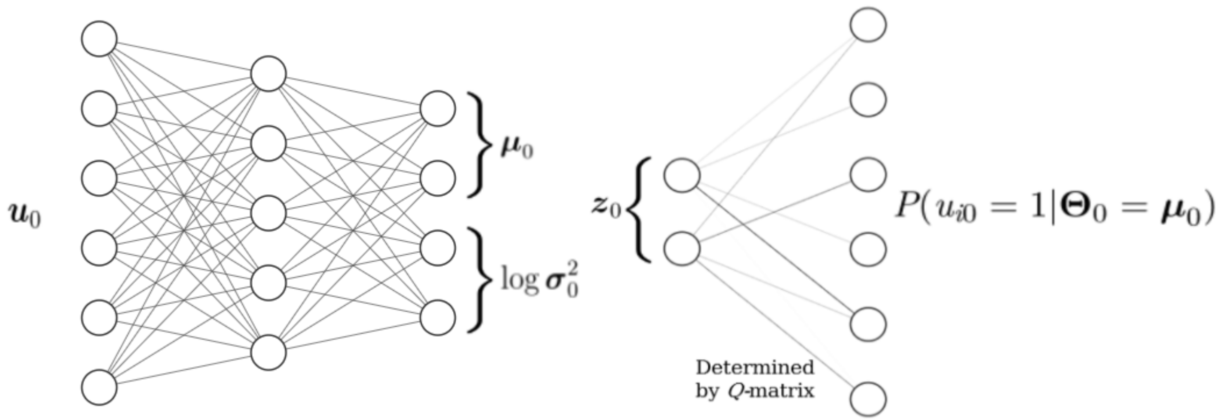
[5] First presented at the International Joint Conference on Neural Networks (IJCNN) 2019

Model Description[5]

- n items $\Rightarrow$ n input/output nodes
- K latent abilities $\Rightarrow$ K-dimensional encoded distribution $\mathcal{N}(0, I)$
- No hidden layers in the VAE decoder
- Restrict nonzero weights in the decoder according to Q-matrix
- Require decoder weights to be nonnegative
  - Avoids reflection $\theta \cdot (-a) = (-\theta) \cdot a$
- Sigmoidal activation function in output layer
- Decoder interpreted as the ML2P model
  - Activation of nodes in encoded layer $\Rightarrow$ latent ability estimates
  - Weights in decoder $\Rightarrow$ discrimination parameter estimates
  - Bias of output nodes $\Rightarrow$ difficulty parameter estimates
  - Output layer $\Rightarrow$ probability of answering items correctly

[5] First presented at the International Joint Conference on Neural Networks (IJCNN) 2019

ML2P-VAE



- Trainable weights in decoder are item parameter estimates
- Feed responses $\boldsymbol{u}_0$ through encoder to obtain ability estimates $\boldsymbol{\Theta}_0 = \boldsymbol{\mu}_0$

Advantages of ML2P-VAE Approach

For the IRT application:

- No trouble for high-dimensional $\boldsymbol{\Theta}$
- Doesn't directly optimize $\boldsymbol{\Theta}$
  - Large number of students isn't a computational burden
- Learning a *function* that maps responses to latent abilities
  - Encoder: $\boldsymbol{u}_0 \mapsto \boldsymbol{\Theta}_0$

Advantages of ML2P-VAE Approach

For the IRT application:

- No trouble for high-dimensional $\mathbf{\Theta}$
- Doesn't directly optimize $\mathbf{\Theta}$
  - Large number of students isn't a computational burden
- Learning a *function* that maps responses to latent abilities
  - Encoder: $\boldsymbol{u}_0 \mapsto \mathbf{\Theta}_0$

In the machine learning field:

- Ability to interpret a hidden neural layer
- Less abstract encoded latent space
- Explainable trainable parameters in decoder

Correlated Latent Traits in IRT

- In real applications, independent skills are not realistic: $\mathbf{\Theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, not $\mathcal{N}(0, I)$.
  - Example: students who are good at addition are also good at subtraction

Correlated Latent Traits in IRT

- In real applications, independent skills are not realistic: $\mathbf{\Theta} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, not $\mathcal{N}(0, I)$.
  - Example: students who are good at addition are also good at subtraction
- Covariance matrix is symmetric, positive definite matrix

$$\Sigma = \begin{bmatrix} \sigma_1^2 & c_{12} & \cdots & c_{1k} \\ c_{21} & \sigma_2^2 & \cdots & c_{2k} \\ \vdots & & \ddots & \vdots \\ c_{k1} & \cdots & c_{k(k-1)} & \sigma_k^2 \end{bmatrix}$$

- With variances σ_i^2 and covariances $c_{ij} = c_{ji}$

Correlated Latent Code in VAE

- In most VAE applications, it is convenient to assume latent code $\boldsymbol{z}$ is *independent*
  - Forces each dimension of $\boldsymbol{z}$ to measure different features
  - $\boldsymbol{z}$ is *abstract*, with **no real-world understanding**

Correlated Latent Code in VAE

- In most VAE applications, it is convenient to assume latent code $\boldsymbol{z}$ is *independent*
  - Forces each dimension of $\boldsymbol{z}$ to measure different features
  - $\boldsymbol{z}$ is *abstract*, with **no real-world understanding**
- For ML2P-VAE, we know that latent code $\boldsymbol{z}$ approximates latent traits $\boldsymbol{\Theta}$
  - We may have **domain knowledge** of the distribution of $\boldsymbol{\Theta}$

KL-Divergence for Multivariate Gaussians

- KL-Divergence between two K-dimensional multivariate Gaussian distributions:

$$\mathcal{D}_{KL}\left[\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) || \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)\right] =$$
$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - K + \ln\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

KL-Divergence for Multivariate Gaussians

- KL-Divergence between two K-dimensional multivariate Gaussian distributions:

$$\mathcal{D}_{KL}\left[\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0) || \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)\right] =$$
$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_1^{-1}\Sigma_0) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \Sigma_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - K + \ln\left(\frac{\det \Sigma_1}{\det \Sigma_0}\right)\right)$$

- When fitting a VAE, $\mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ is assumed to be known, so $\boldsymbol{\mu}_1$ and Σ_1 are constant
- $\boldsymbol{\mu}_0$ and Σ_0 obtained from feeding one sample through the encoder

Implementation Requirements for Correlated VAE

1. KL Divergence calculation uses $\boldsymbol{\mu}_0$, Σ_0, and $\ln\det\Sigma_0$
2. Sample from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:

Implementation Requirements for Correlated VAE

1. KL Divergence calculation uses $\boldsymbol{\mu}_0$, Σ_0, and $\ln \det \Sigma_0$
   - Require $\det \Sigma_0 > 0$ for <u>any</u> input $\boldsymbol{u}_0$
   - Σ_0 is a function of the input $\boldsymbol{u}_0$ and every encoder weight
2. Sample from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:

Implementation Requirements for Correlated VAE

1. KL Divergence calculation uses $\boldsymbol{\mu}_0$, Σ_0, and $\ln\det\Sigma_0$
   - Require $\det\Sigma_0 > 0$ for any input $\boldsymbol{u}_0$
   - Σ_0 is a function of the input $\boldsymbol{u}_0$ and every encoder weight
2. Sample from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$:
   - Find a matrix G such that $GG^T = \Sigma_0$
   - Sample $\boldsymbol{\varepsilon} = (\varepsilon_1, ..., \varepsilon_k)^T$ with each $\varepsilon_i \sim \mathcal{N}(0, 1)$
   - Generate sample $\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + G\boldsymbol{\varepsilon}$

Correlated VAE Implementation[6]

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular

[6] Published in *Machine Learning*, 2021

Correlated VAE Implementation[6]

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular
- Sampling: Calculate $G_0 = e^{L_0}$
  - Note G_0 is lower triangular, nonsingular
  - Send sample $\boldsymbol{z} = \boldsymbol{\mu}_0 + G_0\boldsymbol{\varepsilon}$ through decoder

[6] Published in *Machine Learning*, 2021

Correlated VAE Implementation[6]

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular
- Sampling: Calculate $G_0 = e^{L_0}$
  - Note G_0 is lower triangular, nonsingular
  - Send sample $\boldsymbol{z} = \boldsymbol{\mu}_0 + G_0\boldsymbol{\varepsilon}$ through decoder
- KL Divergence: Calculate $\Sigma_0 = G_0 G_0^T$

[6] Published in *Machine Learning*, 2021

Correlated VAE Implementation[6]

- Architecture: Encoder outputs $K + K(K+1)/2$ nodes
  - K nodes for $\boldsymbol{\mu}_0$, and $K(K+1)/2$ nodes for L_0 lower triangular
- Sampling: Calculate $G_0 = e^{L_0}$
  - Note G_0 is lower triangular, nonsingular
  - Send sample $\boldsymbol{z} = \boldsymbol{\mu}_0 + G_0\boldsymbol{\varepsilon}$ through decoder
- KL Divergence: Calculate $\Sigma_0 = G_0 G_0^T$
  - Claim: Σ_0 is has positive determinant and is symmetric positive definite

[6] Published in *Machine Learning*, 2021

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

Set $\Sigma_0 := G_0 G_0^T$. Now for any nonzero $\boldsymbol{y} \in \mathbb{R}^k$,

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

Set $\Sigma_0 := G_0 G_0^T$. Now for any nonzero $\boldsymbol{y} \in \mathbb{R}^k$,

$$\langle \Sigma_0 \boldsymbol{y}, \boldsymbol{y} \rangle = \boldsymbol{y}^T \Sigma_0 \boldsymbol{y} = \boldsymbol{y}^T G_0 G_0^T \boldsymbol{y} = \langle G_0^T \boldsymbol{y}, G_0^T \boldsymbol{y} \rangle = || G_0^T \boldsymbol{y} ||_2^2 > 0$$

Correlated VAE Implementation

Theorem

Let L_0 be any lower triangular matrix. Then $\Sigma_0 = e^{L_0} \cdot \left(e^{L_0}\right)^\top$ is symmetric, positive definite, and has positive determinant.

Proof.

Consider the matrix exponential

$$G_0 := e^{L_0} = \sum_{n=0}^{\infty} \frac{L_0^n}{n!} = I + L_0 + \frac{1}{2} L_0^2 + \cdots$$

G_0 is lower triangular, since addition and multiplication preserve lower triangular. G_0 is also nonsingular:

$$\det G_0 = \det e^{L_0} = e^{\operatorname{tr} L_0} \neq 0$$

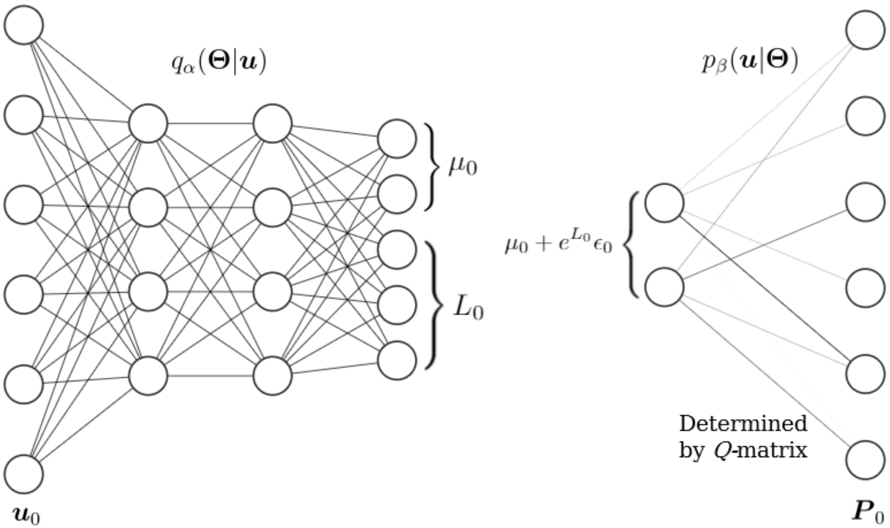
Set $\Sigma_0 := G_0 G_0^T$. Now for any nonzero $\boldsymbol{y} \in \mathbb{R}^k$,

$$\langle \Sigma_0 \boldsymbol{y}, \boldsymbol{y} \rangle = \boldsymbol{y}^T \Sigma_0 \boldsymbol{y} = \boldsymbol{y}^T G_0 G_0^T \boldsymbol{y} = \langle G_0^T \boldsymbol{y}, G_0^T \boldsymbol{y} \rangle = || G_0^T \boldsymbol{y} ||_2^2 > 0$$

Further,

$$\det \Sigma_0 = \det \left(G_0 G_0^\top \right) = \det G_0 \cdot \det G_0^\top = e^{\operatorname{tr} L_0} \cdot e^{\operatorname{tr} L_0} > 0$$

VAE architecture for correlated latent traits



Encoder structure for a VAE fit to latent space $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$

Comparison of IRT Parameter Estimation Methods

- MH-RM
- MC-EM
- QMC-EM

Comparison of IRT Parameter Estimation Methods

- MH-RM
- MC-EM
- QMC-EM
- ML2P-VAE$_{full}$
  - Assume full knowledge of correlation matrix Σ_1
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \Sigma_1)$

Comparison of IRT Parameter Estimation Methods

- MH-RM
- MC-EM
- QMC-EM
- ML2P-VAE$_{full}$
  - Assume full knowledge of correlation matrix Σ_1
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \Sigma_1)$
- ML2P-VAE$_{est}$
  - Unknown correlation matrix $\Sigma_1 \Rightarrow$ estimate it with $\tilde{\Sigma}_1$
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \tilde{\Sigma}_1)$

Comparison of IRT Parameter Estimation Methods

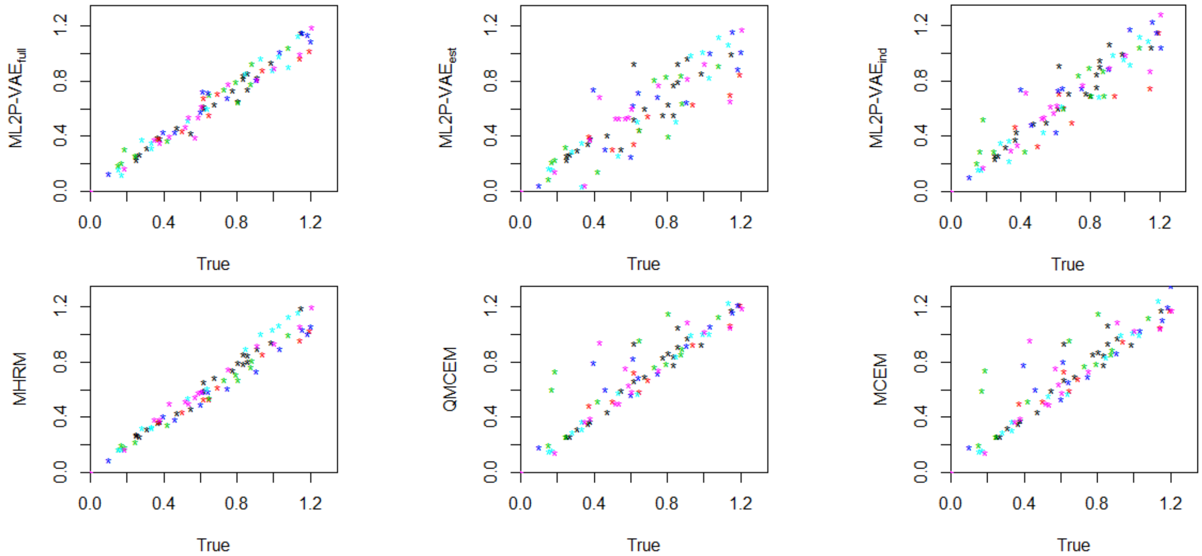
- MH-RM
- MC-EM
- QMC-EM
- ML2P-VAE$_{full}$
  - Assume full knowledge of correlation matrix Σ_1
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \Sigma_1)$
- ML2P-VAE$_{est}$
  - Unknown correlation matrix $\Sigma_1 \Rightarrow$ estimate it with $\tilde{\Sigma}_1$
  - Fit VAE with $\mathcal{N}(\mathbf{0}, \tilde{\Sigma}_1)$
- ML2P-VAE$_{ind}$
  - Unknown correlation matrix $\Sigma_1 \Rightarrow$ assume independent $\boldsymbol{\Theta}$
  - Fit VAE with $\mathcal{N}(\mathbf{0}, I)$

Datasets

	Items	Skills	Students
ECPE	28	3	2,922
Sim-6	50	6	20,000
Sim-20	200	20	50,000
Sim-4	27	4	3,000

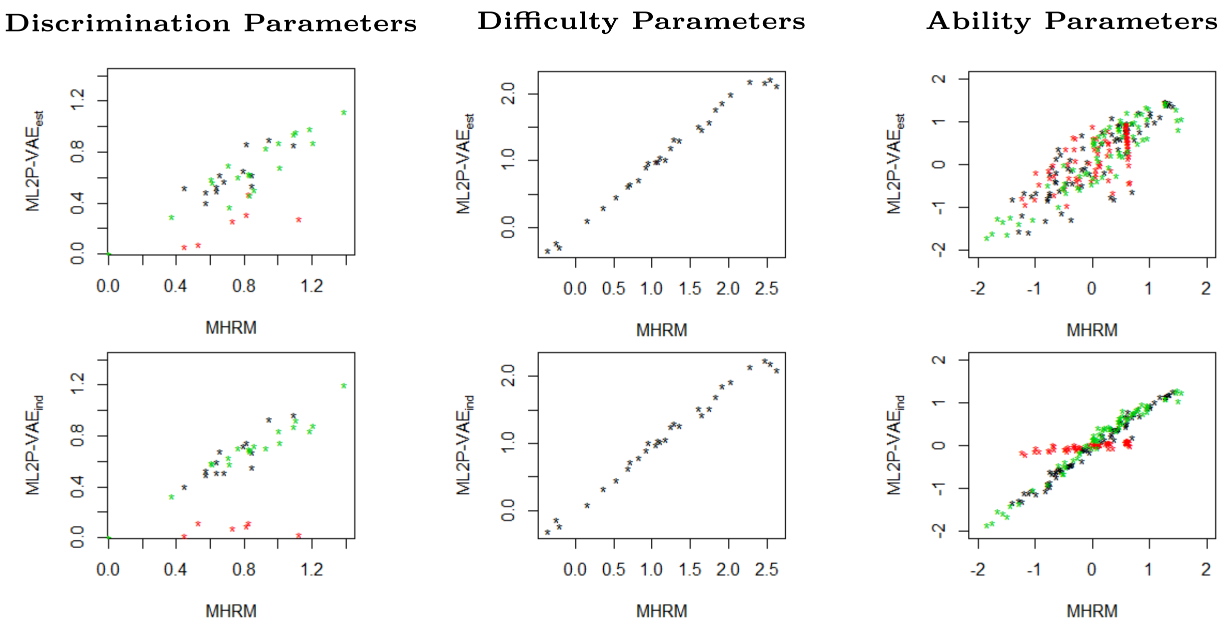
- ECPE is a real-world dataset – no true parameter values
- Sim-6 and Sim-20 have randomly and highly correlated latent skills
- Sim-4 has smaller and more particular correlations

Sim-6 Discrimination Parameter Estimates



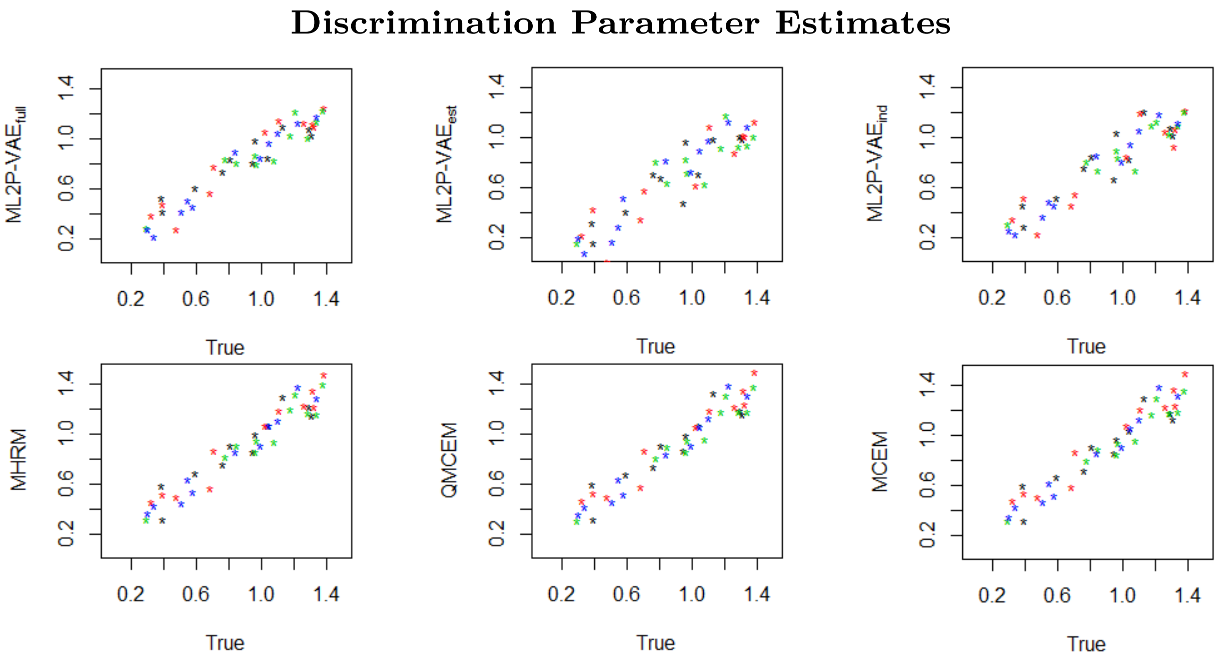
Correlation plots of discrimination parameter estimates – 50 items assessing 6 latent traits.

ECPE Parameter Estimates



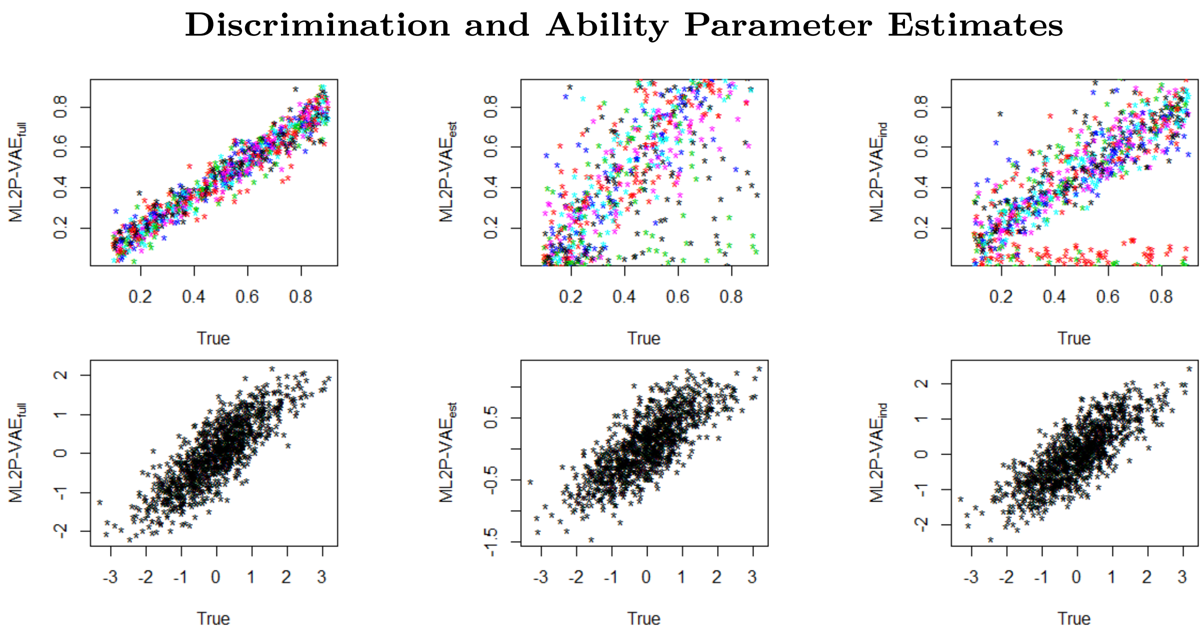
Correlation plots of item and student parameters – 28 items assessing 3 latent traits. No true values – compare against “accepted” MHRM estimates.

Sim-4 Discrimination Parameter Estimates



Correlation plots for discrimination parameters – 27 items assessing 4 latent skills.

Sim-20 Parameter Estimates



Correlation plots for item and student parameters – 200 items assessing 20 latent traits. Traditional methods fail to return estimates on assessments of this size.

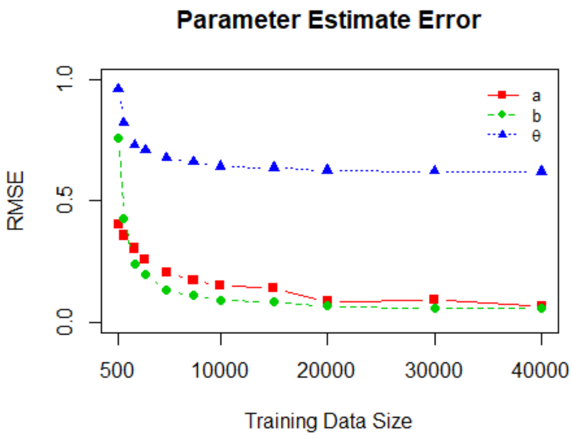
Correlated ML2P-VAE Results

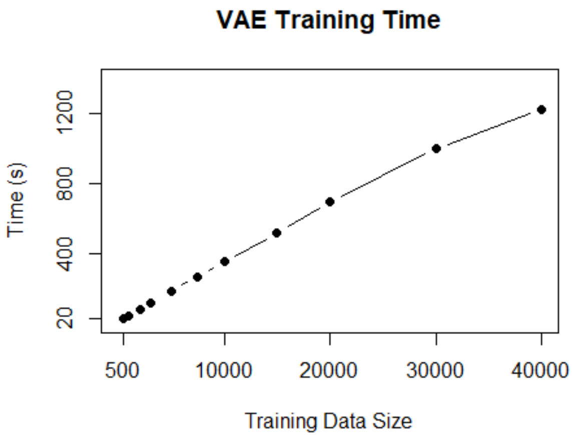
Data Set	Method	***a***.RMSE	***a***.BIAS	***a***.COR	***b***.RMSE	***b***.BIAS	***b***.COR
(i) 6 abilities Sim-6	MHRM	0.0693	0.0319	0.9986	0.0256	-0.0021	0.9999
	QMCEM	0.149	-0.067	0.9939	0.0376	-0.002	0.9998
	MCEM	0.1497	-0.0633	0.9936	0.0383	0.0035	0.9997
	ML2P-VAE$_{full}$	0.0705	0.0255	0.9985	0.0471	-0.0079	0.9996
	ML2P-VAE$_{est}$	0.1803	0.0871	0.9891	0.064	-0.0131	0.9993
	ML2P-VAE$_{ind}$	0.1218	-0.0004	0.9944	0.0597	-0.0145	0.9994
(ii) 3 abilities ECPE	MHRM*	0*	0*	1*	0*	0*	1*
	QMCEM	0.0159	0.0035	0.9999	0.0067	-0.0005	1
	MCEM	0.0228	0.0148	0.9998	0.0064	-0.0008	1
	ML2P-VAE$_{full}$	N/A	N/A	N/A	N/A	N/A	N/A
	ML2P-VAE$_{est}$	0.2794	0.2152	0.9713	0.148	0.0951	0.993
	ML2P-VAE$_{ind}$	0.3208	0.2184	0.9504	0.154	0.0872	0.9932
(iii) 20 abilities Sim-20	MHRM	N/A	N/A	N/A	N/A	N/A	N/A
	QMCEM	N/A	N/A	N/A	N/A	N/A	N/A
	MCEM	N/A	N/A	N/A	N/A	N/A	N/A
	ML2P-VAE$_{full}$	0.078	0.0473	0.9983	0.0608	0.0054	0.9996
	ML2P-VAE$_{est}$	0.2992	-0.1304	0.9822	0.1655	0.1215	0.9987
	ML2P-VAE$_{ind}$	0.2043	0.0592	0.9792	0.0958	-0.0029	0.9992
(iv) 4 abilities Sim-4	MHRM	0.0953	-0.0158	0.9966	0.0614	-0.0101	0.9988
	QMCEM	0.0938	-0.0160	0.9967	0.0614	-0.0179	0.9989
	MCEM	0.0951	-0.0138	0.9966	0.0644	-0.0199	0.9987
	ML2P-VAE$_{full}$	0.1326	0.0780	0.9960	0.0872	-0.0311	0.9978
	ML2P-VAE$_{est}$	0.2526	0.2106	0.9883	0.1035	-0.0337	0.9980
	ML2P-VAE$_{ind}$	0.1658	0.1099	0.9939	0.0944	-0.0254	0.9976

Correlated ML2P-VAE Results

Data Set	Method	Θ.RMSE	Θ.BIAS	Θ.COR	Runtime
(i) 6 abilities Sim-6	MHRM	0.714	-0.0033	0.7006	1110s
	QMCEM	0.7206	0.0023	0.6939	322s
	MCEM	0.7206	-0.0016	0.6938	1009s
	ML2P-VAE$_{full}$	0.6649	-0.0178	0.7476	343s
	ML2P-VAE$_{est}$	0.7109	0.0772	0.7082	364s
	ML2P-VAE$_{ind}$	0.7222	0.0316	0.6928	252s
(ii) 3 abilities ECPE	MHRM*	0*	0*	1*	162s
	QMCEM	0.0111	0.0007	0.9999	33s
	MCEM	0.0132	0.0026	0.9998	192s
	ML2P-VAE$_{full}$	N/A	N/A	N/A	N/A
	ML2P-VAE$_{est}$	0.443	-0.0628	0.8237	61s
	ML2P-VAE$_{ind}$	0.3063	0.01	0.9017	49s
(iii) 20 abilities Sim-20	MHRM	N/A	N/A	N/A	N/A
	QMCEM	N/A	N/A	N/A	N/A
	MCEM	N/A	N/A	N/A	N/A
	ML2P-VAE$_{full}$	0.6145	0.0065	0.7893	1292s
	ML2P-VAE$_{est}$	0.7364	-0.0276	0.7257	961s
	ML2P-VAE$_{ind}$	0.7054	0.0747	0.7135	850s
(iv) 4 abilities Sim-4	MHRM	0.6325	0.0118	0.7697	94s
	QMCEM	0.6326	0.0154	0.7696	29s
	MCEM	0.6326	0.0150	0.7696	196s
	ML2P-VAE$_{full}$	0.6384	0.0210	0.7648	37s
	ML2P-VAE$_{est}$	0.6897	-0.0256	0.7182	38s
	ML2P-VAE$_{ind}$	0.6474	-0.0397	0.7579	30s

Scalability of ML2P-VAE





How does the size of data affect the performance of ML2P-VAE methods?

ML2Pvae Package in R

- Software package on CRAN[7] for easy implementation of ML2P-VAE methods
  - For IRT researchers – requires no knowledge of neural networks or TensorFlow

[7] https://cran.r-project.org/web/packages/ML2Pvae

ML2Pvae Package in R

- Software package on CRAN[7] for easy implementation of ML2P-VAE methods
    - For IRT researchers – requires no knowledge of neural networks or TensorFlow
- Package functions:
    - Construct ML2P-VAE model to desired architecture
        - Independent latent traits or full covariance matrix
    - Wrapper function to train neural network
    - Simple functions to obtain parameter estimates after training

[7] https://cran.r-project.org/web/packages/ML2Pvae

Extending ML2P-VAE to other IRT models

- 3-parameter logistic model uses a "guessing" parameter:

$$P(u_{ij} = 1 | \mathbf{\Theta}_j; \boldsymbol{a}_i, b_i, c_i) = c_i + \frac{1 - c_i}{1 + \exp\left[-\boldsymbol{a}_i^\top \mathbf{\Theta}_j + b_i\right]}$$

Extending ML2P-VAE to other IRT models

- 3-parameter logistic model uses a "guessing" parameter:

$$P(u_{ij} = 1 | \boldsymbol{\Theta}_j; \boldsymbol{a}_i, b_i, c_i) = c_i + \frac{1 - c_i}{1 + \exp\left[-\boldsymbol{a}_i^\top \boldsymbol{\Theta}_j + b_i\right]}$$

- Samejima's graded response model allows for partial credit:

$$P(u_{ij} \geq \frac{0}{3} | \boldsymbol{\Theta}_j; \boldsymbol{a}_{0i}, b_{0i}) = 1$$

$$P(u_{ij} \geq \frac{1}{3} | \boldsymbol{\Theta}_j; \boldsymbol{a}_{1i}, b_{1i}) = \frac{1}{1 + \exp\left[-\boldsymbol{a}_{1i}^\top \boldsymbol{\Theta}_j + b_{1i}\right]}$$

$$P(u_{ij} \geq \frac{2}{3} | \boldsymbol{\Theta}_j; \boldsymbol{a}_{2i}, b_{2i}) = \frac{1}{1 + \exp\left[-\boldsymbol{a}_{2i}^\top \boldsymbol{\Theta}_j + b_{2i}\right]}$$

$$P(u_{ij} = \frac{3}{3} | \boldsymbol{\Theta}_j; \boldsymbol{a}_{3i}, b_{3i}) = \frac{1}{1 + \exp\left[-\boldsymbol{a}_{3i}^\top \boldsymbol{\Theta}_j + b_{3i}\right]}$$

Other Application Areas

IRT has been applied to other psychometric applications outside of education

- Beck Depression Inventory (BDI)
  - 21 item self-reported assessment to diagnose depression
  - Could incorporate other non-response features of respondents (e.g. gender, age)

[8] Work presented at Conference on Fuzzy Systems and Data Mining (FSDM), 2019

Other Application Areas

IRT has been applied to other psychometric applications outside of education

- Beck Depression Inventory (BDI)
  - 21 item self-reported assessment to diagnose depression
  - Could incorporate other non-response features of respondents (e.g. gender, age)
- Personalilty Questionnaires
  - Big Five, Myers-Briggs
  - Incorporate Samejima's graded response model for items on a Likert scale

[8] Work presented at Conference on Fuzzy Systems and Data Mining (FSDM), 2019

Other Application Areas

IRT has been applied to other psychometric applications outside of education

- Beck Depression Inventory (BDI)
  - 21 item self-reported assessment to diagnose depression
  - Could incorporate other non-response features of respondents (e.g. gender, age)
- Personalilty Questionnaires
  - Big Five, Myers-Briggs
  - Incorporate Samejima's graded response model for items on a Likert scale
- Sports analytics and player evaluation[8]
  - Latent athletic ability influences measurable in-game performance

[8] Work presented at Conference on Fuzzy Systems and Data Mining (FSDM), 2019

Knowledge Tracing[9]

- In online learning environments, students have many items available to help learn material
  - AI tutoring systems can suggest which items to present next
- Dynamically track student's knowledge as they progress through questions

[9] Corbett and Anderson, 1995

Knowledge Tracing[9]

- In online learning environments, students have many items available to help learn material
  - AI tutoring systems can suggest which items to present next
- Dynamically track student's knowledge as they progress through questions
- Given a sequence of student interactions (q_t, c_t)
  - q_t indicates the item answered
  - $c_t \in \{0, 1\}$ indicates correctness

[9] Corbett and Anderson, 1995

Knowledge Tracing[9]

- In online learning environments, students have many items available to help learn material
  - AI tutoring systems can suggest which items to present next
- Dynamically track student's knowledge as they progress through questions
- Given a sequence of student interactions (q_t, c_t)
  - q_t indicates the item answered
  - $c_t \in \{0, 1\}$ indicates correctness
- Goal: predict the probability of success on the next item:

$$p_{t+1} = P(c_{t+1} = 1 | (q_1, c_1), \ldots, (q_t, c_t), (q_t, ?))$$

[9] Corbett and Anderson, 1995

Knowledge Tracing[9]

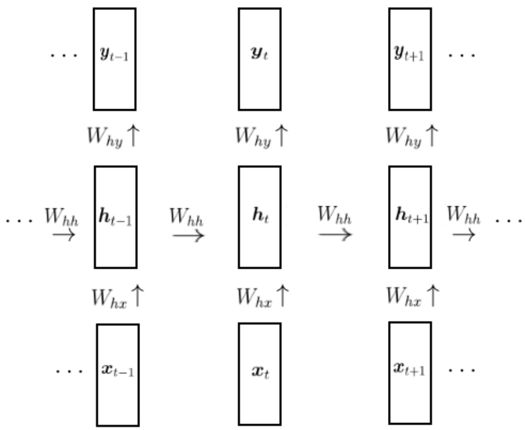
- In online learning environments, students have many items available to help learn material
  - AI tutoring systems can suggest which items to present next
- Dynamically track student's knowledge as they progress through questions
- Given a sequence of student interactions (q_t, c_t)
  - q_t indicates the item answered
  - $c_t \in \{0, 1\}$ indicates correctness
- Goal: predict the probability of success on the next item:

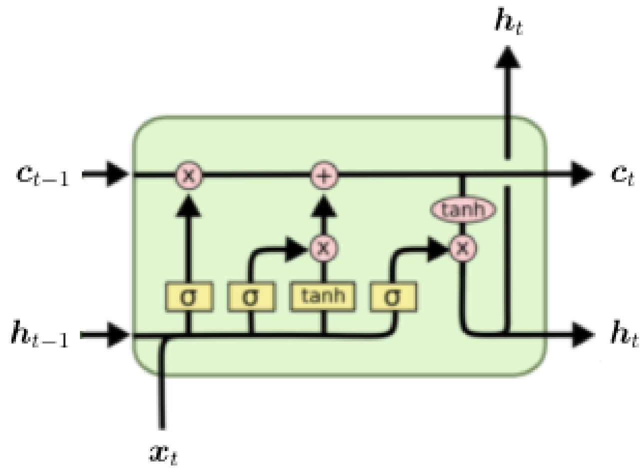
$$p_{t+1} = P(c_{t+1} = 1 | (q_1, c_1), \ldots, (q_t, c_t), (q_t, ?))$$

- Most recent methods use neural networks, similar to NLP application

[9] Corbett and Anderson, 1995

RNN and LSTM





$$\boldsymbol{h}_t = \tanh\left(W_{hh}\boldsymbol{h}_{t-1} + W_{hx}\boldsymbol{x}_t + \boldsymbol{b}_h\right)$$

$$\boldsymbol{f}_t = \sigma(W_f[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}])$$

$$\boldsymbol{a}_t = \left(\sigma(W_u[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]) \times \tanh(W_a[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}])\right)$$

$$\boldsymbol{c}_t = \boldsymbol{c}_{t-1} \times \boldsymbol{f}_t + \boldsymbol{a}_t$$

$$\boldsymbol{h}_t = \tanh(\boldsymbol{c}_t) \times \sigma(W_h[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}])$$

- In NLP, $\boldsymbol{x}_t$ represents a word
- The sequence $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ represents a sentence

Attention Networks[10]

- Given sequence of T observations $\boldsymbol{x}_t \in \mathbb{R}^d$, calculate queries, keys, and values:

$$\boldsymbol{q}_t = W^Q \boldsymbol{x}_t, \quad \boldsymbol{k}_t = W^K \boldsymbol{x}_t, \quad \boldsymbol{v}_t = W^V \boldsymbol{x}_t$$

- Organize into matrices $Q, K, V \in \mathbb{R}^{d \times T}$

[10] Vaswani et al., 2017

Attention Networks[10]

- Given sequence of T observations $\boldsymbol{x}_t \in \mathbb{R}^d$, calculate queries, keys, and values:

$$\boldsymbol{q}_t = W^Q \boldsymbol{x}_t, \quad \boldsymbol{k}_t = W^K \boldsymbol{x}_t, \quad \boldsymbol{v}_t = W^V \boldsymbol{x}_t$$

- Organize into matrices $Q, K, V \in \mathbb{R}^{d \times T}$
- Calculate correlation between current and other timesteps:

$$\boldsymbol{c}_t = \mathrm{softmax}\left(\frac{K\boldsymbol{q}_t}{\sqrt{d}}\right) \in \mathbb{R}^T$$

[10] Vaswani et al., 2017

Attention Networks[10]

- Given sequence of T observations $\boldsymbol{x}_t \in \mathbb{R}^d$, calculate queries, keys, and values:

$$\boldsymbol{q}_t = W^Q \boldsymbol{x}_t, \quad \boldsymbol{k}_t = W^K \boldsymbol{x}_t, \quad \boldsymbol{v}_t = W^V \boldsymbol{x}_t$$

- Organize into matrices $Q, K, V \in \mathbb{R}^{d \times T}$
- Calculate correlation between current and other timesteps:

$$\boldsymbol{c}_t = \operatorname{softmax}\left(\frac{K \boldsymbol{q}_t}{\sqrt{d}}\right) \in \mathbb{R}^T$$

- Attention given as $\boldsymbol{a}_t = V \boldsymbol{c}_t \in \mathbb{R}^d$,

$$A = \operatorname{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) \cdot V \in \mathbb{R}^{T \times d}$$

[10] Vaswani et al., 2017

Attention Networks[10]

- Given sequence of T observations $\boldsymbol{x}_t \in \mathbb{R}^d$, calculate queries, keys, and values:

$$\boldsymbol{q}_t = W^Q \boldsymbol{x}_t, \quad \boldsymbol{k}_t = W^K \boldsymbol{x}_t, \quad \boldsymbol{v}_t = W^V \boldsymbol{x}_t$$

- Organize into matrices $Q, K, V \in \mathbb{R}^{d \times T}$
- Calculate correlation between current and other timesteps:

$$\boldsymbol{c}_t = \mathrm{softmax}\left(\frac{K\boldsymbol{q}_t}{\sqrt{d}}\right) \in \mathbb{R}^T$$

- Attention given as $\boldsymbol{a}_t = V\boldsymbol{c}_t \in \mathbb{R}^d$,

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) \cdot V \in \mathbb{R}^{T \times d}$$

- Feed each $\boldsymbol{a}_t$ through feed-forward network to obtain contextual representation $\boldsymbol{f}_t = FFN(\boldsymbol{a}_t)$

[10] Vaswani et al., 2017

From NLP to Knowledge Tracing

- NLP parallels:
  - Predictive text $\iff$ Predict success on next item
  - One sentence $\iff$ One student's sequence of responses
  - One word $\iff$ One question + response interaction

From NLP to Knowledge Tracing

- NLP parallels:
  - Predictive text $\iff$ Predict success on next item
  - One sentence $\iff$ One student's sequence of responses
  - One word $\iff$ One question + response interaction
- n distinct questions available to student
- Interaction at time t given as tuple (q_t, c_t)
  - $q_t \in \{0, 1, \ldots, n\}$ indexes an item
  - $c_t \in \{0, 1\}$ indicates correctness
  - $\Rightarrow 2n + 1$ possible interactions

From NLP to Knowledge Tracing

- NLP parallels:
  - Predictive text $\iff$ Predict success on next item
  - One sentence $\iff$ One student's sequence of responses
  - One word $\iff$ One question + response interaction
- n distinct questions available to student
- Interaction at time t given as tuple (q_t, c_t)
  - $q_t \in \{0, 1, \ldots, n\}$ indexes an item
  - $c_t \in \{0, 1\}$ indicates correctness
  - $\Rightarrow 2n + 1$ possible interactions
- Trainable embedding matrix $W_e \in \mathbb{R}^{d\times(2n+1)}$
  - Even (resp. odd) columns give d-dimensional representation $\boldsymbol{x}_t$ of answering an item correctly (resp. incorrectly)

DKT and SAKT

- Deep Knowledge Tracing (DKT)[11]
  - Use RNN or LSTM to approximate $P(c_{t+1} = 1|\text{evidence})$
    $$p_{t+1} = \text{FFN}(\text{LSTM}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_1))$$

[11] Piech et al., 2015

[12] Pandey and Karypis, 2019

DKT and SAKT

- Deep Knowledge Tracing (DKT)[11]
  - Use RNN or LSTM to approximate $P(c_{t+1} = 1|\text{evidence})$

$$p_{t+1} = \text{FFN}(\text{LSTM}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_1))$$

- Self-Attentive Knowledge Tracing (SAKT)[12]
  - Use attention networks to approximate $P(c_{t+1} = 1|\text{evidence})$
  - Attention between interactions quantifies similarity between items

$$p_{t+1} = \text{FFN}(\text{Attention}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots, \boldsymbol{x}_1))$$

[11] Piech et al., 2015

[12] Pandey and Karypis, 2019

Do deep models actually "trace" knowledge?

- Best performing deep models: DKT, SAKT, DKVMN[13]
  - Input (q_t, c_t) and q_{t+1}, output p_{t+1}

[13] Zhang et al., 2017

Do deep models actually “trace” knowledge?

- Best performing deep models: DKT, SAKT, DKVMN[13]
  - Input (q_t, c_t) and q_{t+1}, output p_{t+1}
  - Not interpretable
  - No explicit quantification of “knowledge”
    - Post hoc methods to quantify skill mastery

[13] Zhang et al., 2017

Do deep models actually "trace" knowledge?

- Best performing deep models: DKT, SAKT, DKVMN[13]
  - Input (q_t, c_t) and q_{t+1}, output p_{t+1}
  - Not interpretable
  - No explicit quantification of "knowledge"
    - Post hoc methods to quantify skill mastery
- How can we utilize deep methods, while explicitly tracing student knowledge?

[13] Zhang et al., 2017

Do deep models actually "trace" knowledge?

- Best performing deep models: DKT, SAKT, DKVMN[13]
  - Input (q_t, c_t) and q_{t+1}, output p_{t+1}
  - Not interpretable
  - No explicit quantification of "knowledge"
    - Post hoc methods to quantify skill mastery
- How can we utilize deep methods, while explicitly tracing student knowledge?
  - Multidimensional Item Response Theory already provides a framework for computing $P(u_{ij} = 1)$

[13] Zhang et al., 2017

IRT-inspired Knowledge Tracing[14]

- IRT already provides models of the probability of correct response
  - Utilize this in computing p_{t+1}

[14] First presented at the Conference on Artificial Intelligence in Education (AIED), 2021

IRT-inspired Knowledge Tracing[14]

- IRT already provides models of the probability of correct response
  - Utilize this in computing p_{t+1}
- Inject domain-knowledge of Q-matrix and IRT into knowledge tracing framework
  - Output n nodes, representing probability of success on each item
  - Second-to-last layer has K nodes representing K skills
  - Similar to ML2P-VAE, restrict non-zero weights in final layer according to Q-matrix

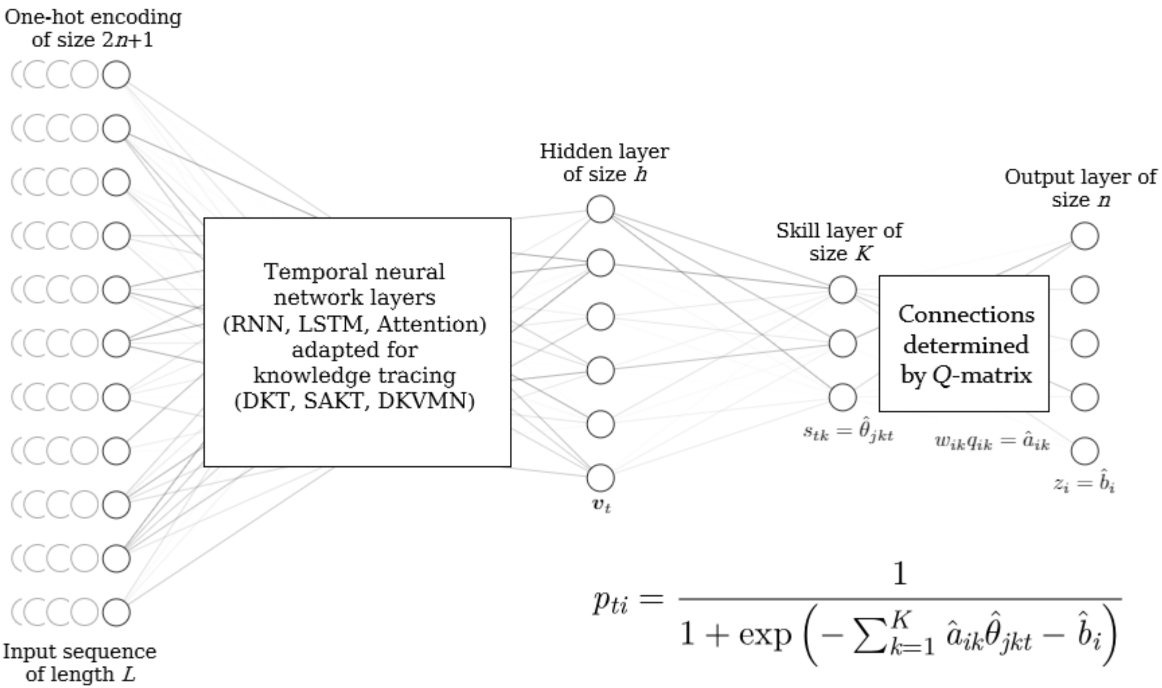
[14] First presented at the Conference on Artificial Intelligence in Education (AIED), 2021

IRT-inspired Knowledge Tracing[14]

- IRT already provides models of the probability of correct response
  - Utilize this in computing p_{t+1}
- Inject domain-knowledge of Q-matrix and IRT into knowledge tracing framework
  - Output n nodes, representing probability of success on each item
  - Second-to-last layer has K nodes representing K skills
  - Similar to ML2P-VAE, restrict non-zero weights in final layer according to Q-matrix
- Knowledge tracing model + IRT parameter estimation method
- Presents trade-off between explainability and predictive power

[14] First presented at the Conference on Artificial Intelligence in Education (AIED), 2021

IRT-inspired Knowledge Tracing



$$p_{ti} = \frac{1}{1 + \exp\left(-\sum_{k=1}^{K} \hat{a}_{ik}\hat{\theta}_{jkt} - \hat{b}_i\right)}$$

Datasets

Dataset	Items	Skills	Students	Interactions
Synth5	50	5	4,000	20K
Sim200	200	20	50,000	10M
Statics2011	987	61	316	135K
Assist2017	4,117	102	1,709	392K

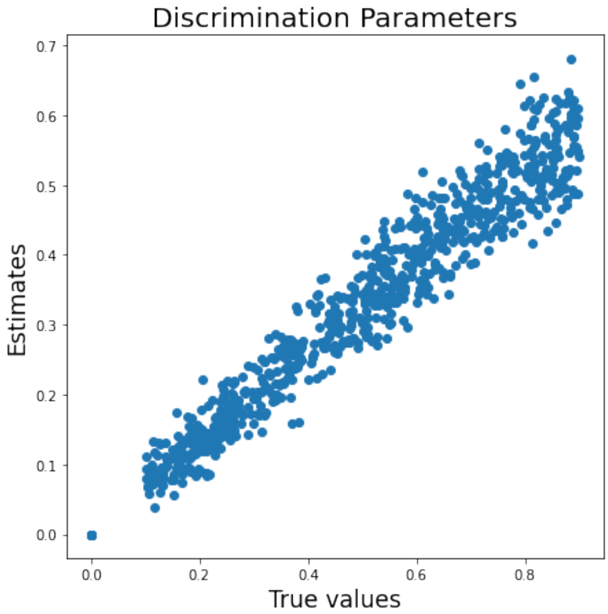
- Synth5 is generated by Rasch model with guessing
  - Difficulty parameters are publicly available
- Sim200 has all item and student parameters available
- Statics2011 and Assist2017 have no true parameters

Results

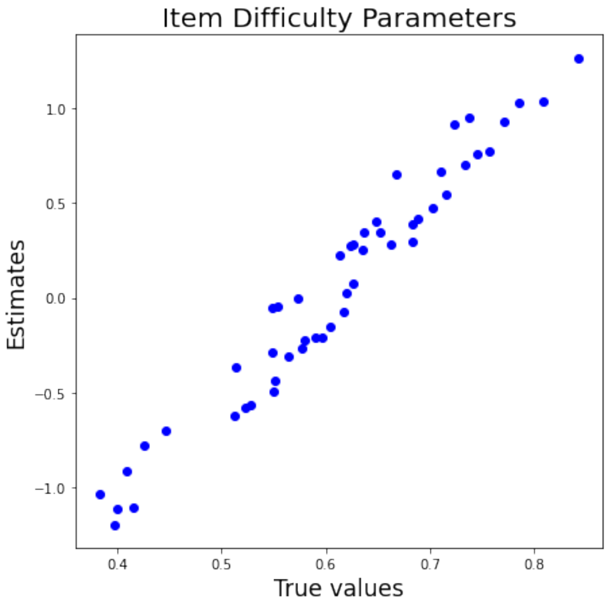
Method	Synth5	Sim200	Statics2011	Assist2017
DKT	0.803	0.838	0.793	0.731
SAKT	0.801	0.834	0.791	0.754
DKVMN	0.827	0.829	0.805	0.796
DKT-IRT	0.799	0.824	0.777	0.724
SAKT-IRT	0.798	0.833	0.775	0.728

Test AUC values for various models on each dataset.

Recovery of IRT parameters

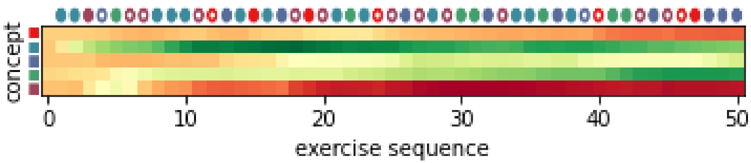


Discrimination parameters from Sim200 data



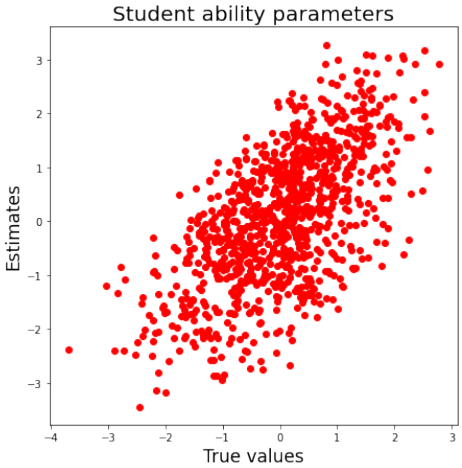
Difficulty parameters from Synth5

Tracing Student Knowledge



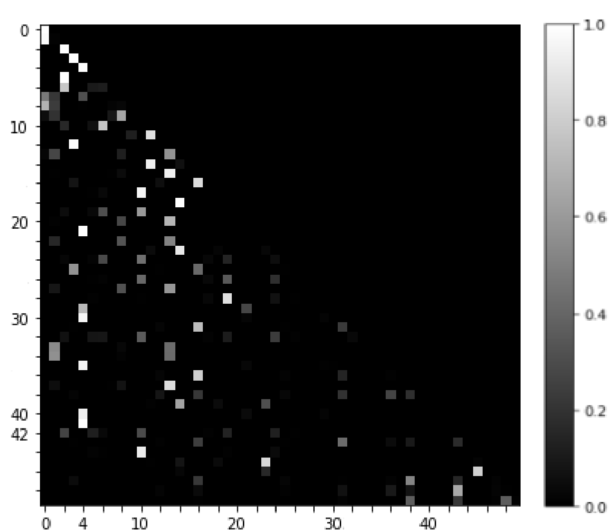
DKT-IRT tracing of student knowledge

SAKT-IRT tracing of student knowledge

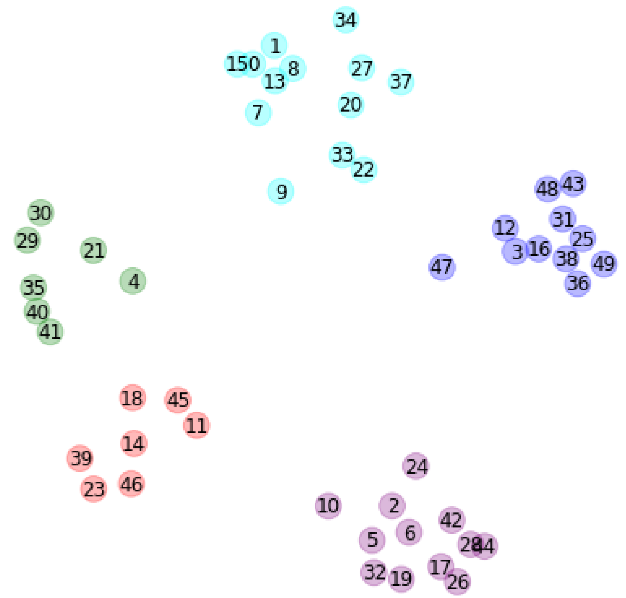


$\boldsymbol{\Theta}$ estimates taken from the final timestep

Learning a Q-matrix



Heatmap of attention weights between Synth5 items



Graph clustering of Synth5 items of similar skill

Utilizing more domain knowledge

- Use Q-matrix in attention calculation
  - Mask out previous interactions from unrelated skills

$$\text{score}_{i,t} = \begin{cases} \dfrac{\boldsymbol{k}_i^\top \boldsymbol{q}_t}{\sqrt{d}} & \text{if interaction } i \text{ shares a skill} \\ & \text{with interaction } t \\ & \\ -\infty & \text{otherwise} \end{cases}$$

Utilizing more domain knowledge

- Use Q-matrix in attention calculation
  - Mask out previous interactions from unrelated skills

$$\text{score}_{i,t} = \begin{cases} \dfrac{\boldsymbol{k}_i^\top \boldsymbol{q}_t}{\sqrt{d}} & \text{if interaction } i \text{ shares a skill} \\ & \text{with interaction } t \\ & \\ -\infty & \text{otherwise} \end{cases}$$

- Extend embedding of interactions to static assessments
  - Deal with unanswered items (missing data)

Summary

- ML2P-VAE
  - Variational autoencoder for confirmatory IRT
- IRT-inspired knowledge tracing
  - Uses ML2P model to predict probability of the next success

Summary

- ML2P-VAE
  - Variational autoencoder for confirmatory IRT
  - Competitive with traditional methods
  - Scalable: capable of estimating high-dimensional $\boldsymbol{\Theta}$
- IRT-inspired knowledge tracing
  - Uses ML2P model to predict probability of the next success

Summary

- ML2P-VAE
  - Variational autoencoder for confirmatory IRT
  - Competitive with traditional methods
  - Scalable: capable of estimating high-dimensional $\boldsymbol{\Theta}$
  - Novel VAE architecture for correlated latent code
  - Interpretable hidden layer and trainable weights
- IRT-inspired knowledge tracing
  - Uses ML2P model to predict probability of the next success

Summary

- ML2P-VAE
  - Variational autoencoder for confirmatory IRT
  - Competitive with traditional methods
  - Scalable: capable of estimating high-dimensional $\mathbf{\Theta}$
  - Novel VAE architecture for correlated latent code
  - Interpretable hidden layer and trainable weights
- IRT-inspired knowledge tracing
  - Uses ML2P model to predict probability of the next success
  - Explicit tracing of student knowledge
  - Recover IRT item parameters
  - Interpretability vs. accuracy trade-off

Thank you!

- My committee: Suely, Mariana, Jonathan, David, and Colleen
- The Mathematics and Computer Science departments

Thank you!

- My committee: Suely, Mariana, Jonathan, David, and Colleen
- The Mathematics and Computer Science departments
- ACT – specifically Scott and Yuchi
- FineTune Learning

Thank you!

- My committee: Suely, Mariana, Jonathan, David, and Colleen
- The Mathematics and Computer Science departments
- ACT – specifically Scott and Yuchi
- FineTune Learning
- My parents: Ken and Karna
- The rest of my family and friends
- My wife: Danielle

Thank you!

- My committee: Suely, Mariana, Jonathan, David, and Colleen
- The Mathematics and Computer Science departments
- ACT – specifically Scott and Yuchi
- FineTune Learning
- My parents: Ken and Karna
- The rest of my family and friends
- My wife: Danielle
    - . . . and our cat Margot

References

- da Silva, Liu, Huggins-Manley, Bazan. “Incorporating the Q-matrix into Multidimensional Item Response Models.” *Journal of Educational and Psychological Measurement*, 2018.
- Baker and Kim. “Item Response Theory: Parameter Estimation Techniques.” CRC Press, 2004.
- Chen et al. “Joint Maximum Likelihood Estimation for high-dimsensional Exploratory Item Factor Analysis.” *Psychometrika*, 84: 124–146, 2019.
- Bock and Aitken. “Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm.” *Psychometrika*, 1981.
- Li Cai. “Metropolis-Hastings Robins-Monro Algorithm for Confirmatory Item Factor Analysis.” *Journal of Educational and Behavioral Statistics*, 35: 307–335, 2010.
- Q. Guo, M. Cutumisu, Y. Cui. “A Neural Network Approach to Estimate Student Skill Mastery in Cognitive Diagnostic Assessments.” In: 10th International Conference on Educational Data Mining. 2017.
- Corbett and Anderson. “Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge.” *User Modeling and User-Adapted Interaction*, 4: 253–278, 1995.
- Vaswani et al. “Attention is All You Need.” *Advances in Neural Information Processing Systems*, 5998–6008, 2017.
- Piech et al. “Deep Knowledge Tracing.” *Advances in Neural Information Processing Systems*, 505–513, 2015.
- Pandey and Karypis. “A Self-Attentive Model for Knowledge Tracing.” *International Conference on Educational Data Mining*, 2019.
- Zhang et al. “Dynamic Key-Value Memory Networks for Knowledge Tracing.” *26th international World Wide Wedb Conference*, 765–774, 2017.

References

- Curi, Converse, Hajewski, Oliveira. "Interpretable Variational Autoencoders for Cognitive Models." In Proceedings of the International Joint Conference on Neural Networks (IJCNN), 2019.
- Converse, Curi, Oliveira. "Autoencoders for Educational Assessment." In Proceedings of the Conference on Artifical Intelligence in Education (AIED), 2019.
- Converse, Curi, Oliveira, Arnold. "Variational Autoencoders for Baseball Player Evaluation." In Proceedings of the Fuzzy Systems and Data Mining Conference (FSDM), 2019.
- Geoffrey Converse. "*ML2Pvae*: Variational Autoencoder Models for IRT Parameter Estimation." R package version 1.0.0. The Comprehensive R Archive Network (CRAN), 2020.
- Converse, Curi, Oliveira, Templin. "Estimation of Multidimensional Item Response Theory Models with Correlated Latent Variables using Variational Autoencoders." *Machine Learning*, 110: 1463–1480, 2021.
- Converse, Oliveira, Pu. "Incorporating Item Response Theory into Knowledge Tracing." In Proceedings of the Conference on Artificial Intelligence in Education (AIED), 2021.
- Pu, Converse, Huang. "Deep Performance Factors Analysis for knowledge Tracing." In Proceedings of the Conference on Artificial Intelligence in Education (AIED), 2021.

Neural Network Methods for Application in Educational Measurement

Geoffrey Converse

University of Iowa

July 15, 2021