Solving Inverse Problems by VAE-like Approaches

Jie Wang
Fourth Year Undergraduate Student
Pure Mathematics Major
Network Coding Lab

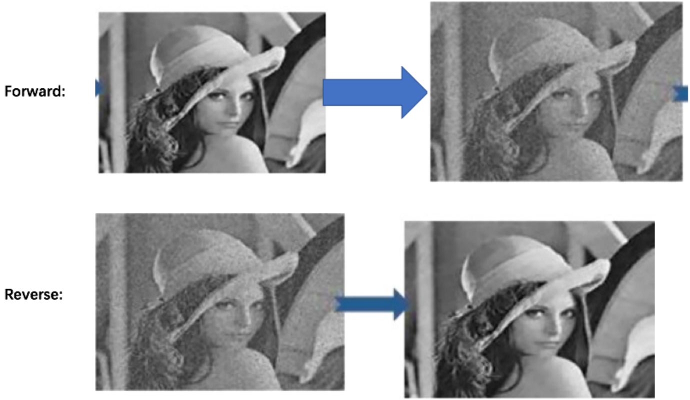
December 4, 2019

Table of Content

Table of Contents

Motivation

- Inverse problems refer to the *reverse process* of a forward problem.
- Example:



Inverse Problem in Statistics

- $\boldsymbol{z} \sim g(\boldsymbol{z}; \boldsymbol{\Lambda})$ with the unknown parameter $\boldsymbol{\Lambda}$
- $\boldsymbol{x}$ is generated through a known likelihood model $p(\boldsymbol{x} \mid \boldsymbol{z})$:
- Given n observed data points $\{\boldsymbol{x}_i\}_{i=1}^n$, recover $\{\boldsymbol{z}_i\}_{i=1}^n$.

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \xrightarrow{\text{Estimator}} \boxed{\{\hat{\boldsymbol{z}}_i\}_{i=1}^n}$$

- Example: $\boldsymbol{z} \sim g(\boldsymbol{z}; \boldsymbol{\Lambda})$, and $\boldsymbol{x} \mid \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}, \sigma^2 \boldsymbol{I})$.

  If given observations $\{\boldsymbol{x}_i\}_{i=1}^n$, then what is $\{\boldsymbol{z}_i\}_{i=1}^n$?

  1. Naive idea: $\boldsymbol{z}_i \approx \boldsymbol{x}_i$
  2. Non-trivial idea:
     - order $\{\boldsymbol{x}_i\}_{i=1}^n$ from smallest to largest, say $\{\boldsymbol{x}_{(i)}\}_{i=1}^n$
     - $\hat{\boldsymbol{z}}_i = \boldsymbol{x}_{(i)}$

Inverse Problem in Statistics

- $\boldsymbol{z} \sim g(\boldsymbol{z}; \boldsymbol{\Lambda})$ with the unknown parameter $\boldsymbol{\Lambda}$
- $\boldsymbol{x}$ is generated through a known likelihood model $p(\boldsymbol{x} \mid \boldsymbol{z})$:
- Given n observed data points $\{\boldsymbol{x}_i\}_{i=1}^n$, recover $\{\boldsymbol{z}_i\}_{i=1}^n$.

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \xrightarrow{\text{Estimator}} \boxed{\{\hat{\boldsymbol{z}}_i\}_{i=1}^n}$$

- Example: $\boldsymbol{z} \sim g(\boldsymbol{z}; \boldsymbol{\Lambda})$, and $\boldsymbol{x} \mid \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}, \sigma^2 \boldsymbol{I})$.

  If given observations $\{\boldsymbol{x}_i\}_{i=1}^n$, then what is $\{\boldsymbol{z}_i\}_{i=1}^n$?

  1. Naive idea: $\boldsymbol{z}_i \approx \boldsymbol{x}_i$
  2. Non-trivial idea:
     - ★ order $\{\boldsymbol{x}_i\}_{i=1}^n$ from smallest to largest, say $\{\boldsymbol{x}_{(i)}\}_{i=1}^n$
     - ★ $\hat{\boldsymbol{z}}_i = \boldsymbol{x}_{(i)}$

Table of Contents

Empirical Bayes approach

- Given n observed data points $\{\boldsymbol{x}_i\}_{i=1}^n$, aim to recover $\{\boldsymbol{z}_i\}_{i=1}^n$:

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \xrightarrow{\text{Inferrer}} \boxed{\{\hat{\boldsymbol{z}}_i\}_{i=1}^n}$$

- **Empirical Bayes**:

Estimation:

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \longrightarrow \boxed{\text{Estimator}} \xrightarrow{\hat{g}(\boldsymbol{z}) := g(\boldsymbol{z}; \hat{\boldsymbol{\Lambda}})}$$

Inference:

$$\xrightarrow{\boldsymbol{x}} \boxed{\text{Inferrer } \hat{g}(\boldsymbol{z} \mid \boldsymbol{x})} \xrightarrow{\hat{\boldsymbol{z}}(\boldsymbol{x})}$$

G-modelling for Empirical Bayes

Estimation:

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^{n}} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^{n}} \longrightarrow \boxed{\text{Estimator}} \xrightarrow{\hat{g}(\boldsymbol{z}) := g(\boldsymbol{z}; \hat{\boldsymbol{\Lambda}})}$$

The estimation problem relies on maximizing the marginal likelihood:

$$\hat{\boldsymbol{\Lambda}} = \arg\max_{\boldsymbol{\Lambda}} \sum_{i=1}^{n} \log p(\boldsymbol{x}_i) \triangleq \max \sum_{i=1}^{n} \log \int g(\boldsymbol{z}_i; \boldsymbol{\Lambda}) p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) \, \mathrm{d}\boldsymbol{z}_i \qquad (1)$$

Intractable for complicated prior distribution or high dimension latent space!

G-modelling for Empirical Bayes

Estimation:

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \longrightarrow \boxed{\text{Estimator}} \xrightarrow{\hat{g}(\boldsymbol{z}) := g(\boldsymbol{z}; \hat{\boldsymbol{\Lambda}})}$$

The estimation problem relies on maximizing the marginal likelihood:

$$\hat{\boldsymbol{\Lambda}} = \arg\max_{\boldsymbol{\Lambda}} \sum_{i=1}^{n} \log p(\boldsymbol{x}_i) \triangleq \max \sum_{i=1}^{n} \log \int g(\boldsymbol{z}_i; \boldsymbol{\Lambda}) p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) \, \mathrm{d}\boldsymbol{z}_i \qquad (1)$$

Intractable for complicated prior distribution or high dimension latent space!

Variational Inference Approach

$$\{\boldsymbol{z}_i\}_{i=1}^n \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \{\boldsymbol{x}_i\}_{i=1}^n \xrightarrow{\text{Inferrer}} \{\hat{\boldsymbol{z}}_i\}_{i=1}^n$$

- Jointly perform the estimation and inference task:

$$\log p(\boldsymbol{x};\boldsymbol{\Lambda}) \geq \mathbb{E}_{q_\phi(\boldsymbol{z})}[\log p(\boldsymbol{z};\boldsymbol{\Lambda}) + \log p(\boldsymbol{x} \mid \boldsymbol{z}) - \log q_\phi(\boldsymbol{z})] \triangleq \text{ELBO}(\boldsymbol{x};\boldsymbol{\Lambda},\phi)$$

  where $q_\phi(\boldsymbol{z})$ is the approximation of the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x};\boldsymbol{\Lambda})$.

- The optimization for ELBO relies on *mean-field approximation* technique:
  1. Large suboptimality Gap, and therefore unreliable estimator and inferrer
  2. Non-convexity Landscape with local optimal points
  3. No reparametrization trick! Complicated simplifcation on ELBO

Variational Inference Approach

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \xrightarrow{\text{Inferrer}} \boxed{\{\hat{\boldsymbol{z}}_i\}_{i=1}^n}$$

- Jointly perform the estimation and inference task:

$$\log p(\boldsymbol{x};\boldsymbol{\Lambda}) \geq \mathbb{E}_{q_\phi(\boldsymbol{z})}[\log p(\boldsymbol{z};\boldsymbol{\Lambda}) + \log p(\boldsymbol{x} \mid \boldsymbol{z}) - \log q_\phi(\boldsymbol{z})] \triangleq \text{ELBO}(\boldsymbol{x};\boldsymbol{\Lambda},\phi)$$

  where $q_\phi(\boldsymbol{z})$ is the approximation of the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x};\boldsymbol{\Lambda})$.
- The optimization for ELBO relies on *mean-field approximation* technique:
  1. Large suboptimality Gap, and therefore unreliable estimator and inferrer
  2. Non-convexity Landscape with local optimal points
  3. No reparametrization trick! Complicated simplifcation on ELBO

Table of Contents

Vanilla VAE-like approach

- Maximize the ELBO function via stochastic optimization techniques:

$$(\hat{\mathbf{\Lambda}}, \hat{\boldsymbol{\phi}}) = \arg\max_{\mathbf{\Lambda},\boldsymbol{\phi}} \sum_{i=1}^{n} \mathbb{E}_{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i)}\Big[\log g(\boldsymbol{z}_i;\mathbf{\Lambda}) + \log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) - \log q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)\Big]$$

  where:
  - $g(\boldsymbol{z}_i;\mathbf{\Lambda})$ and $p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$ is known
  - $q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ is the approximation of the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x};\mathbf{\Lambda})$:

$$(\boldsymbol{\mu}_{1:n}, \log \boldsymbol{\sigma}_{1:n}) = \text{Encoder-Neural-Net}_\phi(\boldsymbol{x}_{1:n});$$

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=1}^{n} q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2));$$

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \approx p(\boldsymbol{z} \mid \boldsymbol{x};\Lambda).$$

- Optimize for ϕ: reparametrization trick $z = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \epsilon$ with $\epsilon \sim \mathcal{N}(0,1)$
  Optimize for Λ: parametric optimization techniques

Vanilla VAE-like approach

- Maximize the ELBO function via stochastic optimization techniques:

$$(\hat{\boldsymbol{\Lambda}}, \hat{\boldsymbol{\phi}}) = \arg\max_{\boldsymbol{\Lambda}, \boldsymbol{\phi}} \sum_{i=1}^{n} \mathbb{E}_{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i)} \Big[\log g(\boldsymbol{z}_i; \boldsymbol{\Lambda}) + \log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) - \log q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i) \Big]$$

  where:
  - $g(\boldsymbol{z}_i; \boldsymbol{\Lambda})$ and $p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$ is known
  - $q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ is the approximation of the true posterior $p(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\Lambda})$:

$$(\boldsymbol{\mu}_{1:n}, \log \boldsymbol{\sigma}_{1:n}) = \text{Encoder-Neural-Net}_\phi(\boldsymbol{x}_{1:n});$$

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_{i=1}^{n} q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2));$$

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \approx p(\boldsymbol{z} \mid \boldsymbol{x}; \Lambda).$$

- Optimize for ϕ: reparametrization trick $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$
  Optimize for $\boldsymbol{\Lambda}$: parametric optimization techniques

Vanilla VAE-like approach

Algorithm 1 Algorithm for Vanilla VAE-like approach. All experiments in the paper used the default values $\alpha = 0.000001,\ B = 128,\ n_{\text{critic}} = 30$

Input: $\{\boldsymbol{x}_i\}_{i=1}^{n}$, observations; $p(\boldsymbol{x} \mid \boldsymbol{z})$, generative model; η, learning rate; n_{critic}, the number of iterations of the ϕ update per $\boldsymbol{\Lambda}$ estimation.

Output: $\hat{\boldsymbol{\Lambda}}, \hat{\phi}$: learnt parameters

$\hat{\boldsymbol{\Lambda}}, \hat{\phi} \leftarrow$ initialize parameters
while Adam not converged do
  for $t = 0, \ldots, n_{\text{critic}}$ do
    Sample $\{x_{(i)}\}_{i=1}^{B}$, a batch from the real data
    Sample random noise $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$
    Generate $\{z_{(i)}\}_{i=1}^{B}$ by the reparameriation trick $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$
    Compute the objective function $\tilde{L}_{\boldsymbol{\Lambda},\phi}(\{x_{(i)}\}_{i=1}^{B}, \{z_{(i)}\}_{i=1}^{B})$ and its gradients $\nabla_{\phi}\tilde{L}_{\boldsymbol{\Lambda},\phi}$:

$$\tilde{L}_{\boldsymbol{\Lambda},\phi}(\{x_{(i)}\}_{i=1}^{B}, \{z_{(i)}\}_{i=1}^{B}) \triangleq \sum_{i=1}^{B} -\log(\boldsymbol{z}_{(i)}; \boldsymbol{\Lambda}) - \log p(\boldsymbol{x}_{(i)} \mid \boldsymbol{z}_{(i)}) - \sum_{j} \log|\boldsymbol{\sigma}_{(i),j}| \tag{4a}$$

    Update $\hat{\phi}$ using Adam optimizer
  end for
  Sample $\{x_{(i)}\}_{i=1}^{B}$, a batch from the real data and random noise $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$
  Generate $\{z_{(i)}\}_{i=1}^{B}$ by the reparameriation trick
  Update $\boldsymbol{\Lambda}$ by solving the maximization problem

$$\hat{\boldsymbol{\Lambda}} = \arg\max_{\boldsymbol{\Lambda}} \sum_{i=1}^{B} \log(\boldsymbol{z}_{(i)}; \boldsymbol{\Lambda}) \tag{4b}$$

end while

VAE-like approach with Inverse Autoregressive Flow

- Vanilla VAE-like approach suffices from the inexact approximation posterior.
- Circumvent it by the Inverse Autoregressive Flow trick:

$$\begin{aligned} \boldsymbol{\epsilon}_0 &\sim \mathcal{N}(0, \boldsymbol{I}), \\ (\boldsymbol{\mu}_0, \log(\boldsymbol{\sigma}_0), \boldsymbol{h}) &= \text{Encoder-Neural-Net}(\boldsymbol{x}; \psi) \\ \boldsymbol{z}_0 &= \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \circ \boldsymbol{\epsilon}_0 \end{aligned}$$

Then apply the following transformations for $t = 1, \ldots, T$:

$$\begin{aligned} (\boldsymbol{m}_t, \boldsymbol{s}_t) &= \text{Auto-regressive-Neural-Net}_t(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{h}; \psi) \\ \boldsymbol{\sigma}_t &= \text{sigmoid}(\boldsymbol{s}_t) \\ \boldsymbol{\epsilon}_t &= \boldsymbol{\sigma}_t \circ \boldsymbol{\epsilon}_{t-1} + (1 - \boldsymbol{\sigma}_t) \circ \boldsymbol{m}_t \end{aligned}$$

and finally $\boldsymbol{z} \triangleq \boldsymbol{\epsilon}_t$.

1. increases flexibility of approximation posteriors
2. scales well to a high-dimensional latent space
3. easy-to-implement by using the open source library

VAE-like approach with Inverse Autoregressive Flow

- Vanilla VAE-like approach suffices from the inexact approximation posterior.
- Circumvent it by the Inverse Autoregressive Flow trick:

$$\boldsymbol{\epsilon}_0 \sim \mathcal{N}(0, \boldsymbol{I}),$$
$$(\boldsymbol{\mu}_0, \log(\boldsymbol{\sigma}_0), \boldsymbol{h}) = \text{Encoder-Neural-Net}(\boldsymbol{x}; \psi)$$
$$\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \circ \boldsymbol{\epsilon}_0$$

Then apply the following transformations for $t = 1, \ldots, T$:

$$(\boldsymbol{m}_t, \boldsymbol{s}_t) = \text{Auto-regressive-Neural-Net}_t(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{h}; \psi)$$
$$\boldsymbol{\sigma}_t = \text{sigmoid}(\boldsymbol{s}_t)$$
$$\boldsymbol{\epsilon}_t = \boldsymbol{\sigma}_t \circ \boldsymbol{\epsilon}_{t-1} + (1 - \boldsymbol{\sigma}_t) \circ \boldsymbol{m}_t$$

and finally $\boldsymbol{z} \triangleq \boldsymbol{\epsilon}_t$.

1. increases flexibility of approximation posteriors
2. scales well to a high-dimensional latent space
3. easy-to-implement by using the open source library

VAE-like approach with Inverse Autoregressive Flow

Two things to be modified based on Vanilla VAE approach:

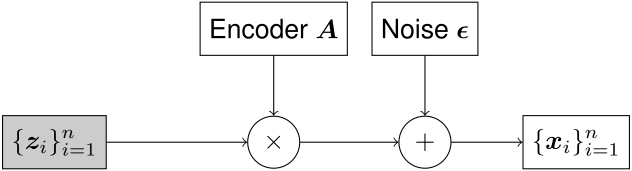
$$(\hat{\boldsymbol{\Lambda}}, \hat{\boldsymbol{\phi}}) = \arg\max_{\boldsymbol{\Lambda},\boldsymbol{\phi}} \sum_{i=1}^{n} \mathbb{E}_{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i)}\left[\log g(\boldsymbol{z}_i; \boldsymbol{\Lambda}) + \log p(\boldsymbol{x}_i \mid \boldsymbol{z}_i) - \log q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)\right]$$

1. The generation of $\boldsymbol{z}$ via $q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$ is from the Inverse Autoregressive Flow process;
2. Substitute the evaluation for the term $\log q_\phi(\boldsymbol{z}_i \mid \boldsymbol{x}_i)$:

$$\log q_\phi(\boldsymbol{z} \triangleq \boldsymbol{\epsilon}_T \mid \boldsymbol{x}) = -\sum_{i=1}^{n}\left(\frac{1}{2}\epsilon_i^2 + \frac{1}{2}\log(2\pi) + \sum_{t=0}^{T}\sigma_{t,i}\right)$$

Table of Contents

Problem Setting



$$
\begin{aligned}
&\text{(Latent Space)} && \boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1}), \quad \boldsymbol{\Lambda} \text{ sparse}, \\
&\text{(Observation Space)} && \boldsymbol{x}_i \mid \boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{A}_{\text{obs}} \boldsymbol{z}_i, \sigma_{\text{obs}}^2 \boldsymbol{I})
\end{aligned}
$$

Vanilla VAE-like Approach

1. Estimation for $\phi := (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$: generate $\boldsymbol{z}_{(i)}$'s and then minimize

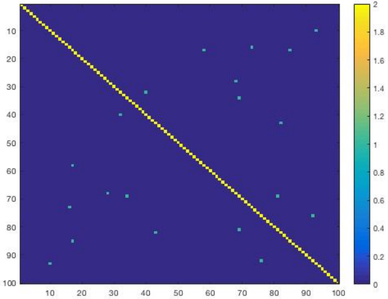
$$\frac{1}{2}\,\text{Trace}\left(\boldsymbol{\Lambda}\cdot\sum_{i=1}^{B}(\boldsymbol{z}_{(i)})(\boldsymbol{z}_{(i)})^{\mathrm{T}}\right)+\frac{1}{2\sigma_{\text{obs}}^2}\sum_{i=1}^{B}\|\boldsymbol{x}_{(i)}-\boldsymbol{A}_{\text{obs}}\boldsymbol{z}_{(i)}\|^2-\sum_{i=1,j}^{B}\log|\sigma_{(i),j}|$$

2. Estimation for $\boldsymbol{\Lambda}$: generate $\boldsymbol{z}_{(i)}$'s and then solve the graphical lasso subproblem:

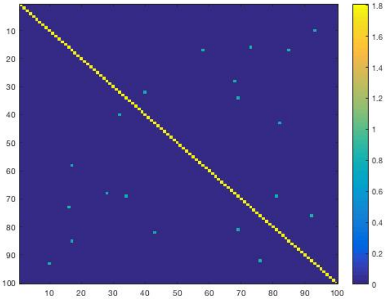
$$\arg\max_{\boldsymbol{\Lambda}}\quad -\lambda\cdot\|\boldsymbol{\Lambda}\|_{\ell_1,\text{off}}+\frac{B}{2}\log|\boldsymbol{\Lambda}|-\frac{B}{2}\,\text{Trace}\left[\boldsymbol{\Lambda}\cdot\frac{1}{B}\sum_{i=1}^{B}(\boldsymbol{z}_{(i)})(\boldsymbol{z}_{(i)})^{\mathrm{T}}\right]$$

Disadvantage: graphical lasso problem is computationally expansive!

Simulation Results



(a) Underlying Precision matrix $\mathbf{\Lambda}_{\text{true}}$



(b) Estimated Precision matrix $\hat{\mathbf{\Lambda}}$

Simulation Results

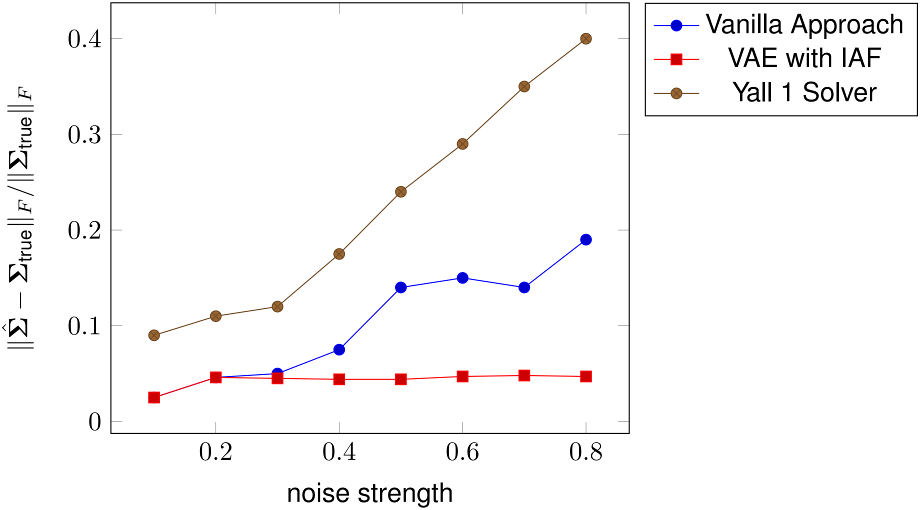


Table of Contents

Future Work

- Consider the signal reconstruction task from Gaussian Mixture Model:

$$\boxed{\{\boldsymbol{z}_i\}_{i=1}^n} \xrightarrow{p(\boldsymbol{x}_i \mid \boldsymbol{z}_i)} \boxed{\{\boldsymbol{x}_i\}_{i=1}^n} \xrightarrow{\text{Estimator}} \boxed{\{\hat{\boldsymbol{z}}_i\}_{i=1}^n}$$

  where

$$\boldsymbol{z} \sim \sum_{i=1}^{K} \pi_i \cdot \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_{(K)}^{-1})$$

- Consider the more general likelihood function $p(\boldsymbol{x} \mid \boldsymbol{z})$. For instance, the inverse problem can be used for solving stochastic differential equation:

$$\boldsymbol{x} = \mathcal{G}(\boldsymbol{z}, \boldsymbol{\epsilon}),$$

  where $\mathcal{G}(\cdot)$ is the generative model, $\boldsymbol{\epsilon}$ is the random noise, and $\boldsymbol{x}$ are observed data points.

Concluding Remarks

- The inverse problem is an old problem studied in history, and we use VAE-like approach to efficiently solve it;
- Such approach is applicable to general prior distribution patterns and likelihood models
- The suboptimality gap for this approach is an open problem.