Rethinking Sparsity-Aware Bayesian Learning for Signal Processing and Machine Learning

Feng Yin, Lei Cheng, Sergios Theodoridis

The Chinese University of Hong Kong (Shenzhen), China
Zhejiang University, China
National and Kapodistrian University of Athens, Greece

2023.06.04

Overview

Outline

Background

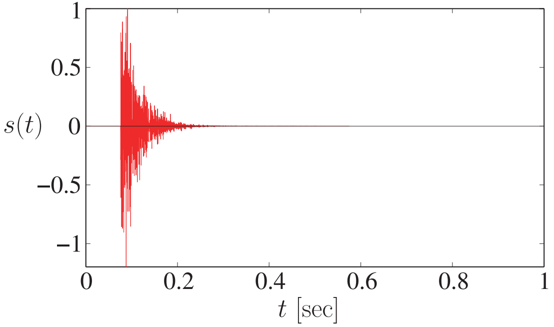
- Sparsity-aware learning has offered novel theoretical tools and solutions to challenging practical problems in various sectors.
- Applications include biomedicine, astronomy, signal processing, wireless communications, and data science.
- Sparsity-aware learning can be achieved from either a Frequentist path or a Bayesian path.

Sparsity-aware learning is capable of

- improving the overall parameter estimation performance (in terms of bias and variance);
- guarding against over-fitting, particularly when data samples are scarce;
- coping with ill-conditions, such as rank-deficient covariance matrices,etc;
- providing a solution to underdetermined linear system of equations;
- generating parsimonious models with sparse and interpretable signal representation.

Classic Examples: Echo Path Representation

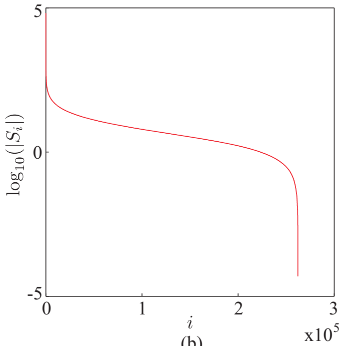
- Sparsity is an attribute that is met in a plethora of natural signals, because nature tends to be parsimonious.
- Echo path vector comprising the values of impulse response samples is sparse.



Impulse response function of an echo path in a telephone network. Short duration and no prior knowledge about its appearance in time.

Classic Examples: Image Compression

(a)



(b)

(a) A 512 × 512 image; and (b) the magnitude of its discrete cosine transform (DCT) components in descending order.

- Heart of compression: More than 95% of the total energy contributed by only 5% of the largest components.

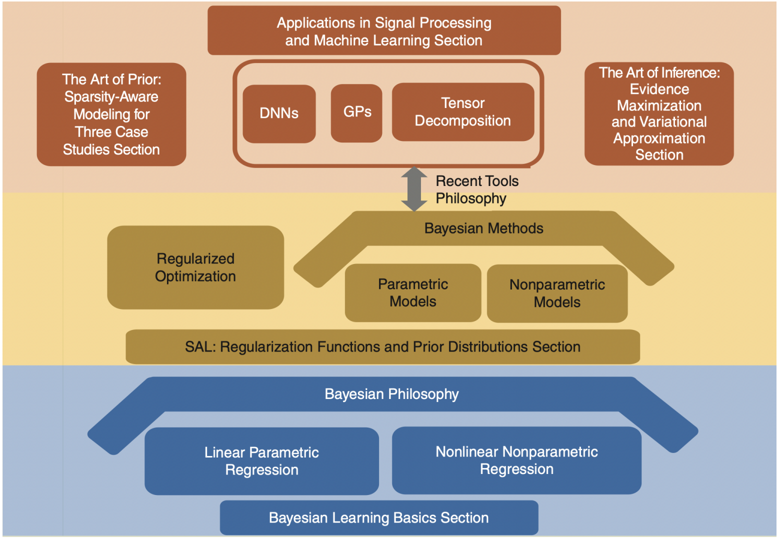
Scope of Tutorial

- This tutorial will focus on sparsity-aware learning via Bayesian path.
- This tutorial will show sparsity-promoting priors and inference strategies for modern Gaussian process, Bayesian neural network, and tensor decomposition models.

Scope of Tutorial

- This tutorial will focus on sparsity-aware learning via Bayesian path.
- This tutorial will show sparsity-promoting priors and inference strategies for modern Gaussian process, Bayesian neural network, and tensor decomposition models.
- This tutorial was prepared mainly based on:
  1. L. Cheng, F. Yin, S. Theodoridis, S. Chatzis and T. -H. Chang, "Rethinking Bayesian Learning for Data Analysis: The art of prior and inference in sparsity-aware modeling," in IEEE Signal Processing Magazine, vol. 39, no. 6, pp. 18-52, Nov. 2022.
  2. Sergios Theodoridis, "Machine Learning: A Bayesian and Optimization Perspective, Academic Press, 2nd Edition, 2020.

Organization



Applications in Signal Processing and Machine Learning Section
The Art of Prior: Sparsity-Aware Modeling for Three Case Studies Section
DNNs
GPs
Tensor Decomposition
The Art of Inference: Evidence Maximization and Variational Approximation Section
Recent Tools Philosophy
Bayesian Methods
Regularized Optimization
Parametric Models
Nonparametric Models
SAL: Regularization Functions and Prior Distributions Section
Bayesian Philosophy
Linear Parametric Regression
Nonlinear Nonparametric Regression
Bayesian Learning Basics Section

Outline

Section Goals

This section aims to

- provide some basics about Bayesian learning philosophy;
- introduce important notations and quantities of Bayesian learning;
- introduce Bayesian parametric and nonparametric models.

Brief Review of Frequentist Path

In the past, more dominantly, sparsity-aware learning was conducted via regularized optimization of the general form:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell(\mathcal{D}; \boldsymbol{\theta}) + \lambda \cdot r(\boldsymbol{\theta}),$$

Brief Review of Frequentist Path

In the past, more dominantly, sparsity-aware learning was conducted via regularized optimization of the general form:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ell(\mathcal{D}; \boldsymbol{\theta}) + \lambda \cdot r(\boldsymbol{\theta}),$$

where

- $\boldsymbol{\theta}$ represents the desired model parameters;
- $\ell(\mathcal{D}; \boldsymbol{\theta})$ is a cost function w.r.t. a finite dataset, $\mathcal{D} \triangleq \{\boldsymbol{y}, \boldsymbol{X}\}$, to measure the data fitting performance;
- $r(\boldsymbol{\theta})$ is a regularization function w.r.t. the model parameters, $\boldsymbol{\theta}$, to steer the sparsity structure embedding;
- λ is a regularization parameter to balance data fitting and sparsity structure embedding.

RIDGE Regression vs. LASSO Regression

- RIDGE regression adopts least-squares loss and ℓ_2 norm as regularization,

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2 \right).$$

- RIDGE regression admits a closed-form solution:

$$\hat{\boldsymbol{\theta}}_R = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}.$$

RIDGE Regression vs. LASSO Regression

- RIDGE regression adopts least-squares loss and ℓ_2 norm as regularization,

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_2 \right).$$

- RIDGE regression admits a closed-form solution:

$$\hat{\boldsymbol{\theta}}_R = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}.$$

- The solution has no sparsity structure.

- LASSO regression adopts least-squares loss but ℓ_1 norm as regularization,

$$\hat{\boldsymbol{\theta}}_L = \arg\min_{\boldsymbol{\theta}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 \right).$$

- LASSO regression adopts least-squares loss but ℓ_1 norm as regularization,

$$\hat{\boldsymbol{\theta}}_L = \arg\min_{\boldsymbol{\theta}} \left((\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}\|_1 \right).$$

- LASSO regression can be equivalently written as

$$\begin{aligned} \hat{\boldsymbol{\theta}}_L = \quad & \arg\min\nolimits_{\boldsymbol{\theta}} (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\ & \text{s.t.} \quad \|\boldsymbol{\theta}\|_1 \le \rho \end{aligned}$$

or (also known as basis pursuit denoising)

$$\begin{aligned} \hat{\boldsymbol{\theta}}_L = \quad & \arg\min\nolimits_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_1 \\ & \text{s.t.} \quad (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \le \epsilon \end{aligned}.$$

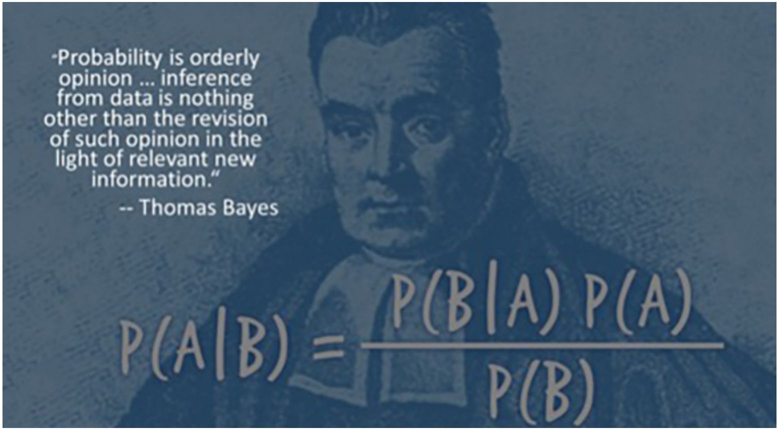
- The solution requires iterative algorithms but shows sparsity structure.

References for Frequentist Path

- Chapter 9 and 10 of *Machine Learning: A Bayesian and Optimization Perspective* by Sergios Theodoridis.
- Chapter 13 of *Machine Learning: A Probabilistic Perspective* by Kevin P. Murphy.
- Various Tutorials given at NeurIPS, ICLR, AAAI,etc, in the past five years.

The Heart of Bayesian Learning

In this tutorial, we will rethink sparsity-aware learning via the Bayesian path paved by the elegant and generic Bayes' Theorem.


"Probability is orderly opinion ... inference from data is nothing other than the revision of such opinion in the light of relevant new information."
-- Thomas Bayes
P(A|B) = P(B|A) P(A) / P(B)

Basics of Bayesian Philosophy

Notations

- Let $\mathcal{D}$ be the observed (training) dataset.
- Let $\mathcal{M}$ be the underlying model for having generated the data.
- Let $\boldsymbol{\theta} \in \mathbb{R}^{L\times 1}$ be the unknown model parameters, treated as random variables.
- Let $\boldsymbol{\theta} \sim p_{\mathcal{M}}(\boldsymbol{\theta}; \boldsymbol{\eta}_p)$ be the *prior distribution* of $\boldsymbol{\theta}$.
- Let $\boldsymbol{\eta}_p$ be a set of deterministic yet unknown *hyperparameters* to specify the prior.
- Let $p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})$ be the *likelihood* to describe the observed data given the values of the parameters.

Bayes' Theorem and Important Quantities

$$p_{\mathcal{M}}(\boldsymbol{\theta}|\mathcal{D};\boldsymbol{\eta}) = \frac{p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)}{p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta})},$$

where

- Prior: $p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)$
- Likelihood: $p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})$
- Posterior: $p_{\mathcal{M}}(\boldsymbol{\theta}|\mathcal{D};\boldsymbol{\eta})$
- Evidence/Marginal Likelihood: $p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta})$

Bayes' Theorem and Important Quantities

$$p_{\mathcal{M}}(\boldsymbol{\theta}|\mathcal{D};\boldsymbol{\eta}) = \frac{p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)}{p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta})},$$

where

- Prior: $p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)$
- Likelihood: $p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})$
- Posterior: $p_{\mathcal{M}}(\boldsymbol{\theta}|\mathcal{D};\boldsymbol{\eta})$
- Evidence/Marginal Likelihood: $p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta})$

- Evidence is computed as the marginalization of the likelihood:

$$p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta}) = \int p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)d\boldsymbol{\theta}.$$

- Classic ML estimator can be obtained via:

$$\hat{\eta}_{ML} = \arg\max_{\boldsymbol{\eta}} \log p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta})$$

Evidence vs. ELBO

- Evidence/Marginal likelihood is not always analytically tractable due to the multiple integrals involved.

Evidence vs. ELBO

- Evidence/Marginal likelihood is not always analytically tractable due to the multiple integrals involved.
- For Bayesian inference, we could alternatively consider the evidence lower bound (ELBO):

$$\log p_{\mathcal{M}}(\mathcal{D}; \boldsymbol{\eta}) \geq \mathcal{L}(q(\boldsymbol{\theta}); \boldsymbol{\eta}) \triangleq \int q(\boldsymbol{\theta}) \log \frac{p_{\mathcal{M}}(\mathcal{D}, \boldsymbol{\theta}; \boldsymbol{\eta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}.$$

Evidence vs. ELBO

- Evidence/Marginal likelihood is not always analytically tractable due to the multiple integrals involved.
- For Bayesian inference, we could alternatively consider the evidence lower bound (ELBO):

$$\log p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta}) \geq \mathcal{L}(q(\boldsymbol{\theta});\boldsymbol{\eta}) \triangleq \int q(\boldsymbol{\theta}) \log \frac{p_{\mathcal{M}}(\mathcal{D},\boldsymbol{\theta};\boldsymbol{\eta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}.$$

- Tightness of ELBO is determined by the closeness between the variational distribution $q(\boldsymbol{\theta})$ and the posterior $p_{\mathcal{M}}(\boldsymbol{\theta}|\mathcal{D};\boldsymbol{\eta})$.
- Solving ELBO maximization problem for different learning models:

$$\max_{q(\boldsymbol{\theta}),\boldsymbol{\eta}} \mathcal{L}(q(\boldsymbol{\theta});\boldsymbol{\eta}).$$

Parametric Model vs. Nonparametric Model

- Parametric Model:
    1. adopts a finite set of parameters $\boldsymbol{\theta}$,
    2. outputs point estimate solely relying on the model parameters trained based on the observed data, $\mathcal{D}$,
    3. Bayesian linear regression model is a represented one.

Parametric Model vs. Nonparametric Model

- Parametric Model:
  1. adopts a finite set of parameters $\boldsymbol{\theta}$,
  2. outputs point estimate solely relying on the model parameters trained based on the observed data, $\mathcal{D}$,
  3. Bayesian linear regression model is a represented one.

- Nonparametric Model:
  1. adopts an infinite dimensional $\boldsymbol{\theta}$, often regarded as a random function,
  2. keeps updating $\boldsymbol{\theta}$ as data $\mathcal{D}$ expands with finer granularity,
  3. Gaussian process model is a represented one.

Parametric Model: Bayesian Linear Regression

We start with the ordinary linear regression model:

$$y_n = f(\boldsymbol{x}_n; \boldsymbol{\theta}) + v_n, \quad n = 1, 2, ..., N.$$

Parametric Model: Bayesian Linear Regression

We start with the ordinary linear regression model:

$$y_n = f(\boldsymbol{x}_n; \boldsymbol{\theta}) + v_n, \quad n = 1, 2, ..., N.$$

where the regression function is

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{x},$$

and

- $\boldsymbol{x} = [x_1, x_2, ..., x_L]^T$ is the input/feature vector of size L,
- $\boldsymbol{\theta}$ is the vector of model parameters following certain distribution,
- $v_n, n = 1, 2, ..., N$ are independent noise terms.

Classically, by assuming:

- Gaussian likelihood, owing to $\{v_n\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(v_n; 0, \beta^{-1})$, where β represents the noise precision, and

$$p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}(y_n; \boldsymbol{\theta}^T \boldsymbol{x}_n, \beta^{-1}).$$

Classically, by assuming:

- Gaussian likelihood, owing to $\{v_n\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(v_n; 0, \beta^{-1})$, where β represents the noise precision, and

$$p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}(y_n; \boldsymbol{\theta}^T \boldsymbol{x}_n, \beta^{-1}).$$

- Gaussian prior on the unknown parameters,

$$p_{\mathcal{M}}(\boldsymbol{\theta}; \boldsymbol{\eta}_p) = \prod_{l=1}^{L} \mathcal{N}(\theta_l; 0, \alpha_l^{-1}),$$

where α_l is the precision for θ_l, and $\boldsymbol{\eta}_p \triangleq [\alpha_1, \alpha_2, \cdots, \alpha_L]^T$.

We can derive then two important quantities:

- Gaussian evidence obtained through marginalization:

$$\begin{aligned} p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta}) &= \int p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)d\boldsymbol{\theta} \\ &= \mathcal{N}(\boldsymbol{y};\mathbf{0},\beta^{-1}\boldsymbol{I}+\boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{X}^T), \end{aligned}$$

where $\boldsymbol{A} \triangleq \mathrm{diag}\{\alpha_1,\alpha_2,\cdots,\alpha_L\}$ and $\boldsymbol{\eta} = [\boldsymbol{\eta}_p^T, \beta]^T$.

We can derive then two important quantities:

- Gaussian evidence obtained through marginalization:

$$\begin{aligned} p_{\mathcal{M}}(\mathcal{D};\boldsymbol{\eta}) &= \int p_{\mathcal{M}}(\mathcal{D}|\boldsymbol{\theta})p_{\mathcal{M}}(\boldsymbol{\theta};\boldsymbol{\eta}_p)d\boldsymbol{\theta} \\ &= \mathcal{N}(\boldsymbol{y};\mathbf{0},\beta^{-1}\boldsymbol{I}+\boldsymbol{X}\boldsymbol{A}^{-1}\boldsymbol{X}^T), \end{aligned}$$

  where $\boldsymbol{A} \triangleq \mathrm{diag}\{\alpha_1,\alpha_2,\cdots,\alpha_L\}$ and $\boldsymbol{\eta} = [\boldsymbol{\eta}_p^T,\beta]^T$.

- Gaussian posterior eventually derived from Bayes' theorem:

$$p_{\mathcal{M}}(\boldsymbol{\theta}|\mathcal{D};\boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{\theta};\boldsymbol{\mu},\boldsymbol{\Sigma}),$$

  where

$$\begin{aligned} \boldsymbol{\mu} &= \beta\boldsymbol{\Sigma}\boldsymbol{X}^T\boldsymbol{y}, \\ \boldsymbol{\Sigma} &= (\boldsymbol{A}+\beta\boldsymbol{X}^T\boldsymbol{X})^{-1}. \end{aligned}$$

Often, point prediction is obtained by $y_* = \boldsymbol{\mu}^T\boldsymbol{x}_*$.

Nonparametric Model: Gaussian Process Regression

A representative example of Bayesian nonparametric model is the Gaussian process model for machine learning with the definitions:

Definition 1: Gaussian process [Rasmussen, 2006][0]

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

[0] C.Rasmussen and C.Williams, Gaussian Process for Machine Learning, MIT Press,2006.

Nonparametric Model: Gaussian Process Regression

A representative example of Bayesian nonparametric model is the Gaussian process model for machine learning with the definitions:

Definition 1: Gaussian process [Rasmussen, 2006]^0

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Definition 2: Gaussian process [Theodoridis, 2020]

A random process, $f(\boldsymbol{x})$, is called a Gaussian process if and only if for any finite number of points, $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$, the associated joint probability density function (pdf), $p(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \cdots, f(\boldsymbol{x}_N))$ is Gaussian.

^{0}C.Rasmussen and C.Williams, Gaussian Process for Machine Learning, MIT Press,2006.

- A real GP, $f(\boldsymbol{x})$, is completely specified by its mean function $m(\boldsymbol{x})$ and its covariance function/kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ as

$$\begin{aligned} m(\boldsymbol{x}) &\triangleq \mathbb{E}[f(\boldsymbol{x})], \\ k\left(\boldsymbol{x}, \boldsymbol{x}'\right) &\triangleq \mathbb{E}\left[\left(f(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(f\left(\boldsymbol{x}'\right) - m\left(\boldsymbol{x}'\right)\right)\right]. \end{aligned}$$

- A real GP, $f(\boldsymbol{x})$, is completely specified by its mean function $m(\boldsymbol{x})$ and its covariance function/kernel function $k(\boldsymbol{x}, \boldsymbol{x'})$ as

$$\begin{aligned} m(\boldsymbol{x}) &\triangleq \mathbb{E}[f(\boldsymbol{x})], \\ k\left(\boldsymbol{x}, \boldsymbol{x}'\right) &\triangleq \mathbb{E}\left[(f(\boldsymbol{x}) - m(\boldsymbol{x}))\left(f\left(\boldsymbol{x}'\right) - m\left(\boldsymbol{x}'\right)\right)\right]. \end{aligned}$$

- We denote a GP as

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(m(\boldsymbol{x}), k\left(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\eta}_p\right)\right).$$

- Model representation power is determined to large extent by the kernel function.

Kernel Functions

- Every specification of a kernel function determines to a family of functions.
- Some elementary kernel functions are:
  1. Linear kernel (with $\boldsymbol{\eta}_p = \sigma_0$)

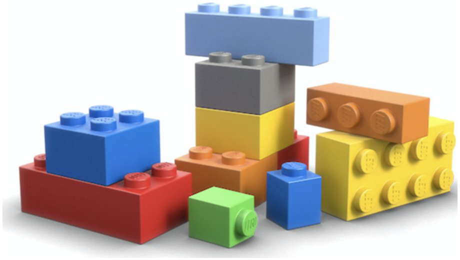
$$k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\eta}_p) = \sigma_0 + \boldsymbol{x}^T \boldsymbol{x}'$$

  2. Squared Exponential (SE)/Gaussian kernel (with $\boldsymbol{\eta}_p = [\sigma_s^2, \ell_s]^T$)

$$k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\eta}_p) = \sigma_s^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\ell_s^2}\right)$$

  3. Matern kernels, rational quadratic kernels, periodic kernels, local periodic kernels, etc.

Kernel Selection and Design



Many elementary kernel functions available

We focus on a class of interpretable and sparsity-promoting kernel functions to form a GP prior.

Gaussian Process for Regression

We consider the following GP regression model

$$y = f(\boldsymbol{x}) + \varepsilon,$$

Gaussian Process for Regression

We consider the following GP regression model

$$y = f(\boldsymbol{x}) + \varepsilon,$$

where

- y is a continuous-valued scalar output;
- the underlying regression function is represented by a zero-mean Gaussian process $f(\boldsymbol{x})$ specified by $k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\eta}_p)$;
- the noise terms are Gaussian i.i.d. with zero-mean and unknown variance β^{-1};
- The set of all unknown parameters, $\boldsymbol{\eta} = [\boldsymbol{\eta}_p, \beta]^T$.

- Given a finite number of training input points: $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$, $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$.
- The selected Gaussian process prior boils down to a multivariate Gaussian distribution, with

$$\boldsymbol{f} := [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)]^T \sim \mathcal{N}\left(\mathbf{0}, K\left(X, X\right)\right).$$

- Given a finite number of training input points: $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$, $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$.
- The selected Gaussian process prior boils down to a multivariate Gaussian distribution, with

$$\boldsymbol{f} := [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)]^T \sim \mathcal{N}(\mathbf{0}, K(X, X)).$$

- The likelihood function, given $\boldsymbol{f}$, is also Gaussian of the form:

$$p(\boldsymbol{y}|\boldsymbol{f}; \boldsymbol{\eta}) = \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{f}, \beta^{-1}\boldsymbol{I}_n\right).$$

- Given a finite number of training input points: $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$, $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$.
- The selected Gaussian process prior boils down to a multivariate Gaussian distribution, with

$$\boldsymbol{f} := [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_n)]^T \sim \mathcal{N}(\mathbf{0}, K(X, X)).$$

- The likelihood function, given $\boldsymbol{f}$, is also Gaussian of the form:

$$p(\boldsymbol{y}|\boldsymbol{f}; \boldsymbol{\eta}) = \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{f}, \beta^{-1}\boldsymbol{I}_n\right).$$

Recall Fundamental Tasks

- Kernel selection/design
- Kernel hyperparameter optimization
- Posterior prediction of novel data points

Task: Kernel Hyperparameter Optimization

- Conventionally, hyperparameters $\boldsymbol{\eta}$, are derived from maximizing the evidence.
- Due to Gaussian process prior and Gaussian likelihood function, the evidence in closed-form can be derived as:

$$p(\boldsymbol{y}; \boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{y}; \mathbf{0}, \boldsymbol{K}(X, X; \boldsymbol{\eta}_p) + \beta^{-1} \boldsymbol{I}).$$

Task: Kernel Hyperparameter Optimization

- Conventionally, hyperparameters $\boldsymbol{\eta}$, are derived from maximizing the evidence.
- Due to Gaussian process prior and Gaussian likelihood function, the evidence in closed-form can be derived as:

$$p(\boldsymbol{y};\boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{0}, \boldsymbol{K}(X, X; \boldsymbol{\eta_p}) + \beta^{-1}\boldsymbol{I}).$$

- Such optimization problem is typically smooth but non-convex for general regression and classification problems.
- Inaccurate prediction will incur when a bad local optimum is found by gradient descent type of methods.

Task: Posterior Prediction

- Goal: To predict $\boldsymbol{y}_* = [y_{*,1}, y_{*,2}, ..., y_{*,n_*}]^T$ given novel (test) inputs $X_* = [\mathbf{x}_{*,1}, \mathbf{x}_{*,2}, ..., \mathbf{x}_{*,n_*}]$ from its posterior distribution $p(\mathbf{y}_*|\mathcal{D}, X_*; \boldsymbol{\eta})$, often short as $p(\mathbf{y}_*|\boldsymbol{y})$.

Task: Posterior Prediction

- Goal: To predict $\boldsymbol{y}_* = [y_{*,1}, y_{*,2}, ..., y_{*,n_*}]^T$ given novel (test) inputs $X_* = [\mathbf{x}_{*,1}, \mathbf{x}_{*,2}, ..., \mathbf{x}_{*,n_*}]$ from its posterior distribution $p(\mathbf{y}_* | \mathcal{D}, X_*; \boldsymbol{\eta})$, often short as $p(\mathbf{y}_* | \boldsymbol{y})$.
- The joint distribution of the training output $\mathbf{y}$ and test output $\mathbf{y}_*$ can be derived as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \beta^{-1}\boldsymbol{I}_n & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) + \beta^{-1}\boldsymbol{I}_{n_*} \end{bmatrix}\right).$$

- Applying the conditional Gaussian results, we obtain:

$$p(\mathbf{y}_*|\boldsymbol{y}) \sim \mathcal{N}\left(\bar{\mathbf{m}}, \bar{V}\right),$$

where

$$\bar{\mathbf{m}} \triangleq K(X_*, X)\left[K(X, X) + \beta^{-1} I_n\right]^{-1} \mathbf{y},$$

$$\begin{aligned}\bar{V} \triangleq & K(X_*, X_*) + \beta^{-1} I_{n_*} \\ & - K(X_*, X)\left[K(X, X) + \beta^{-1} I_n\right]^{-1} K(X, X_*).\end{aligned}$$

- Applying the conditional Gaussian results, we obtain:

$$p(\mathbf{y}_*|\mathbf{y}) \sim \mathcal{N}\left(\bar{\mathbf{m}}, \bar{V}\right),$$

where

$$\bar{\mathbf{m}} \triangleq K(X_*, X)\left[K(X,X) + \beta^{-1} I_n\right]^{-1} \mathbf{y},$$

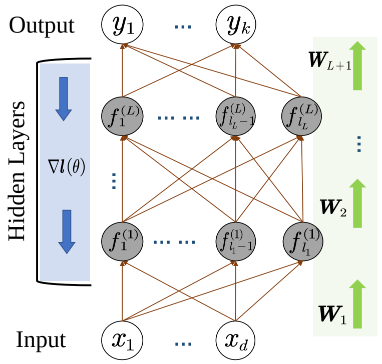
$$\bar{V} \triangleq K(X_*, X_*) + \beta^{-1} I_{n_*} - K(X_*, X)\left[K(X,X) + \beta^{-1} I_n\right]^{-1} K(X, X_*).$$

Interpretations

- The posterior mean can be seen as a linear predictor.
- The posterior variance accounts the **difference** between variance of the prior and variance explained by $\mathcal{D}$.

- When using linear kernel the above results boil down to Bayesian linear regression.

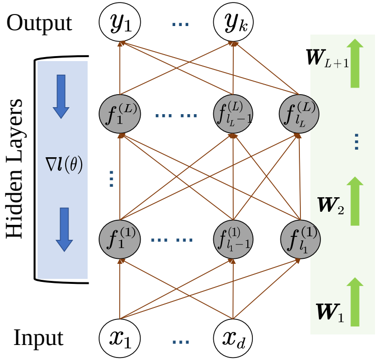
Connections between GP and NNs


Output
y_1 ··· y_k
Hidden Layers
$\nabla l(\theta)$
$f_1^{(L)}$ ··· ··· $f_{l_L-1}^{(L)}$ $f_{l_L}^{(L)}$
$f_1^{(1)}$ ··· ··· $f_{l_1-1}^{(1)}$ $f_{l_1}^{(1)}$
W_{L+1}
W_2
W_1
Input
x_1 ··· x_d

- Each hidden layer comprises a large number of nonliner activation functions to mimic the role of neurons in our brain.

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks", *Neural Networks*, 1991.

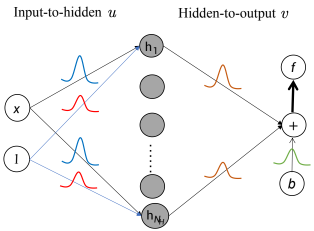
Connections between GP and NNs

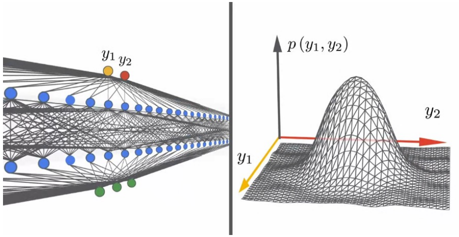


- Each hidden layer comprises a large number of nonliner activation functions to mimic the role of neurons in our brain.
- Neural network (NN) can approximate any smooth function arbitrarily well according to the universal approximation theorem [Hornik'91]. [1]

[1] K. Hornik, "Approximation capabilities of multilayer feedforward networks", *Neural Networks*, 1991.

Connections between GP and NNs





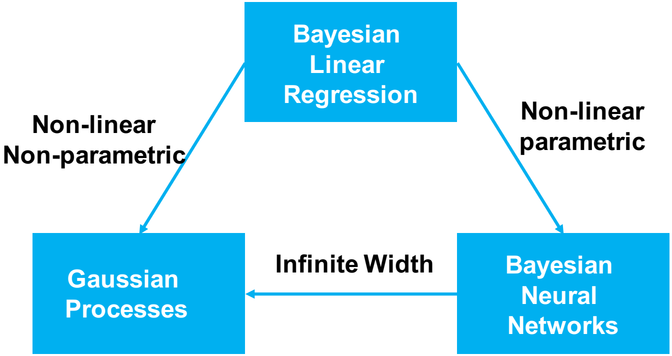
- GP and single layer NN [MacKay'98] [2]
- Neural network kernel

- GP and infinitely wide DNN [Lee'18] [3]
- Neural tangent kernel

[2] D. MacKay, "Introduction to Gaussian processes", *NATO ASI series F computer and systems sciences*, 1998.

[3] J. Lee, *et al.*, "Deep neural networks as Gaussian processes", *ICLR*, 2018.

Section Conclusion



Outline

Section Goals

This section aims to

- show a family of interpretable and sparsity-promoting kernel prior for nonparametric GP model;
- show the kernel hyperparameter optimization process and its benefits;
- explain the sparsity-aware property among other valuable ones.

Linear Multiple Kernel Design

- We focus on the big family of linear multiple kernels:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{Q} \alpha_i k_i(\boldsymbol{x}, \boldsymbol{x}'),$$

  where the weights, $\alpha_i, i = 1, 2, \cdots, Q$, need to be optimized.
- The number of subkernels, Q, is often set large to form an over-complete dictionary of basis.
- Identify the most effective basis subkernels.

- Kernel design can be conducted either in raw input domain or in frequency domain.
- Design in raw input domain by ensembling a number of elementary kernels $\Rightarrow$ analysis-of-variance kernel family.
- Design in frequency domain $\Rightarrow$ sparse spectrum kernel family.

Design in Frequency Domain

Lemma (of Bochner's Theorem)

For 1-D time series where $\mathbf{x} = t$, $\tau = t - t'$, its stationary kernel function, $k(\tau)$, and the spectral density, $S(\omega)$, form a Fourier pair:

$$k(\tau) = \int_{\mathbb{R}^1} S(f) \exp\left[j 2\pi\tau\omega\right] df,$$
$$S(\omega) = \int_{\mathbb{R}^1} k(\tau) \exp\left[-j 2\pi\tau\omega\right] d\tau.$$

Note: for simplicity, we let t represent time and ω represent normalized frequency.

First Sparse Spectrum (SS) Kernel

- Inspired by Bayesian linear regression with an over-complete set of trigononometric basis functions, $\{\cos(2\pi\omega_i x), \sin(2\pi\omega_i x)\}_{i=1}^{Q}$. [4]
- Feature mapping vector $\phi(x)$ contains all Q basis functions.

[4] M. Lázaro-Gredilla, J. Quinonero Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," J. Mach. Learn. Res., vol. 11, pp. 1865-1881, Aug. 2010.

First Sparse Spectrum (SS) Kernel

- Inspired by Bayesian linear regression with an over-complete set of trigononometric basis functions, $\{\cos(2\pi\omega_i x), \sin(2\pi\omega_i x)\}_{i=1}^{Q}$. [4]
- Feature mapping vector $\phi(x)$ contains all Q basis functions.
- Choosing the weights to follow i.i.d. zero mean Gaussian, $\mathcal{N}(0, \frac{\sigma_0^2}{Q})$, the equivalent kernel function:

$$k(x, x') = \frac{\sigma_0^2}{Q}\phi^T(x)\phi(x') = \frac{\sigma_0^2}{Q}\sum_{i=1}^{Q}\cos(2\pi\omega_i(x - x')).$$

- The kernel hyperparameters, $\boldsymbol{\eta}_p = [\sigma_0, \omega_1, \omega_2, ..., \omega_Q]^T$, are to be optimized.

[4] M. Lázaro-Gredilla, J. Quinonero Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," J. Mach. Learn. Res., vol. 11, pp. 1865-1881, Aug. 2010.

Spectral Mixture Kernel: Frequency Domain

- Wilson *et.al.* proposed Spectral Mixture (SM) kernel [5] to approximate the spectral density of the underlying kernel:

$$S(\omega) = \frac{1}{2}\sum_{i=1}^{Q} \frac{\alpha_i}{\sqrt{2\pi\sigma_i^2}} \left\{ \exp\left[\frac{-(\omega-\mu_i)^2}{2\sigma_i^2}\right] + \exp\left[\frac{-(\omega+\mu_i)^2}{2\sigma_i^2}\right] \right\}$$

[5] A. Wilson and R. P. Adams, "Gaussian process kernels for pattern discovery and extrapolation," in Proc. Int. Conf. Mach. Learn. (ICML), Atlanta, GA, USA, 2013, pp. 1067 − 1075.

Spectral Mixture Kernel: Frequency Domain

- Wilson *et.al.* proposed Spectral Mixture (SM) kernel [5] to approximate the spectral density of the underlying kernel:

$$S(\omega) = \frac{1}{2}\sum_{i=1}^{Q} \frac{\alpha_i}{\sqrt{2\pi\sigma_i^2}} \left\{ \exp\left[\frac{-(\omega-\mu_i)^2}{2\sigma_i^2}\right] + \exp\left[\frac{-(\omega+\mu_i)^2}{2\sigma_i^2}\right] \right\}$$

- Q is a preselected large number; α_i, μ_i, σ_i^2 are the weight, mean (center frequency) and variance of the i-th mixture component.

[5] A. Wilson and R. P. Adams, "Gaussian process kernels for pattern discovery and extrapolation," in Proc. Int. Conf. Mach. Learn. (ICML), Atlanta, GA, USA, 2013, pp. 1067 − 1075.

SM Kernel: Raw Input Domain

- Taking the inverse Fourier transform of $S(\omega)$ yields a stationary kernel in the original input domain as

$$k(t, t^{'}; \boldsymbol{\eta}_p) = k(\tau) = \sum_{i=1}^{m} \alpha_i \underbrace{\exp\left[-2\pi^2\tau^2{\sigma_i}^2\right]\cos(2\pi\tau\mu_i)}_{k_i(\tau)}$$

- The kernel hyperparameters $\boldsymbol{\eta}_p = [\alpha_1, \mu_1, \sigma_1, ..., \alpha_Q, \mu_Q, \sigma_Q]^T$ are to be optimized.

SM Kernel: Approximation Capacity

- Approximate any stationary kernel arbitrary well in L_1 norm.

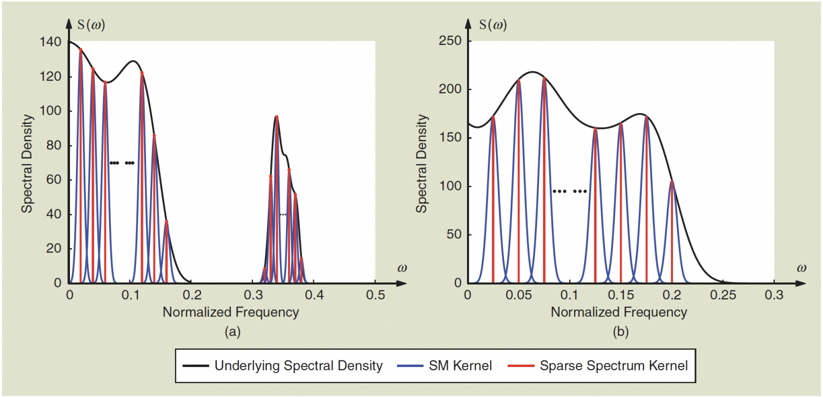
Theorem

For any stationary kernel $\tilde{k}(\tau)$ and an arbitrary $\epsilon > 0$, there exists Q_ϵ and for $m > Q_\epsilon$,

$$\int_{-\infty}^{\infty} \left| \sum_{i=1}^{m} \alpha_i \exp\left[-2\pi^2\tau^2\sigma_i^2\right] \cos(2\pi\tau\mu_i) - \tilde{k}(\tau) \right| d\tau < \epsilon.$$

Proof: Based on Wiener's theorem of approximation.

SS Kernel vs. SM Kernel



- SM kernel employs a mixture of Gaussian basis functions (blue curves)
- SS kernel employs a mixture of Dirac deltas (red vertical lines)

Numerical Difficulties of SS and SM Kernels

- It is generally difficult to tune the hyper-parameters due to large-scale non-convex optimization problem.
- Slower convergence and rather high probability of hitting bad local minimum.
- Relatively high computational time for big dataset.

Grid SM (GSM) Kernel

- To obtain nicer optimization structure, fix the mean and variance parameters to preselected grids, yielding the GSM kernel [6]:

$$k(t, t^{'}; \boldsymbol{\eta}_p) = k(\tau) = \sum_{i=1}^{m} \alpha_i \underbrace{\exp\left[-2\pi^2\tau^2{\sigma_i}^2\right]\cos(2\pi\tau\mu_i)}_{k_i(\tau)}$$

[6] Feng Yin, S. Theodoridis, and Z.-Q. Luo, et.al. "Linear multiple lowrank kernel based stationary Gaussian processes regression for time series," IEEE Trans. Signal Process., vol. 68, pp. 5260-5275, Sep. 2020.

Grid SM (GSM) Kernel

- To obtain nicer optimization structure, fix the mean and variance parameters to preselected grids, yielding the GSM kernel [6]:

$$k(t, t^{'}; \boldsymbol{\eta}_p) = k(\tau) = \sum_{i=1}^{m} \alpha_i \underbrace{\exp\left[-2\pi^2\tau^2{\sigma_i}^2\right]\cos(2\pi\tau\mu_i)}_{k_i(\tau)}$$

- $k_i(\tau)$ can be seen as a fixed sub-kernel without any hyper-parameters to be tuned.
- The kernel hyperparameters $\boldsymbol{\eta}_p = \boldsymbol{\alpha} \triangleq [\alpha_1, \alpha_2, ..., \alpha_m]^T \geq \mathbf{0}$ are to be optimized.

GSM kernel is essentially a linear multiple kernel!

[6] Feng Yin, S. Theodoridis, and Z.-Q. Luo, et.al. "Linear multiple lowrank kernel based stationary Gaussian processes regression for time series," IEEE Trans. Signal Process., vol. 68, pp. 5260-5275, Sep. 2020.

Inference Algorithms for GP with GSM Kernel

- Evidence is mathematically tractable for GP model.
- Maximizing the evidence is equivalent to

$$\hat{\boldsymbol{\eta}}_{ML} = \arg \min_{\boldsymbol{\eta}=\{\boldsymbol{\alpha},\sigma_e^2\}} \ell(\boldsymbol{\eta}) \triangleq \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\alpha}, \sigma_e^2)\boldsymbol{y} + \log\det\left(\boldsymbol{C}(\boldsymbol{\alpha}, \sigma_e^2)\right)$$
$$s.t. \quad \boldsymbol{\alpha} \geq \mathbf{0}, \sigma_e^2 \geq 0.$$

- $\boldsymbol{C}(\boldsymbol{\alpha}, \sigma_e^2) \triangleq \sum_{i=1}^{Q} \alpha_i \boldsymbol{K}_i + \sigma_e^2 \boldsymbol{I}_n$ is the overall covariance matrix including the noise term.

Inference Algorithms for GP with GSM Kernel

- Evidence is mathematically tractable for GP model.
- Maximizing the evidence is equivalent to

$$\hat{\boldsymbol{\eta}}_{ML} = \arg \min_{\boldsymbol{\eta}=\{\boldsymbol{\alpha},\sigma_e^2\}} \ell(\boldsymbol{\eta}) \triangleq \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\alpha}, \sigma_e^2)\boldsymbol{y} + \log\det\left(\boldsymbol{C}(\boldsymbol{\alpha}, \sigma_e^2)\right)$$
$$s.t. \quad \boldsymbol{\alpha} \geq \mathbf{0}, \sigma_e^2 \geq 0.$$

- $\boldsymbol{C}(\boldsymbol{\alpha}, \sigma_e^2) \triangleq \sum_{i=1}^{Q} \alpha_i \boldsymbol{K}_i + \sigma_e^2 \boldsymbol{I}_n$ is the overall covariance matrix including the noise term.
- Let $g(\boldsymbol{\eta}) \triangleq \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\alpha}, \sigma_e^2)\boldsymbol{y}$ and $h(\boldsymbol{\eta}) \triangleq= -\log\det\left(\boldsymbol{C}(\boldsymbol{\alpha}, \sigma_e^2)\right)$.
- The above is provably a difference-of-convex (DCP) problem.

- An effective way to handle DCP is via the majorization-minimization (MM) algorithm.

- An effective way to handle DCP is via the majorization-minimization (MM) algorithm.
- Introduce a so-called majorization function $\bar{\ell}(\boldsymbol{\eta}, \boldsymbol{\eta}^k)$ of $\ell(\boldsymbol{\eta})$ at $\boldsymbol{\eta}^k \in \Theta$ and solve instead

$$\boldsymbol{\eta}^{k+1} = \arg\min_{\boldsymbol{\eta} \in \Theta} \bar{\ell}(\boldsymbol{\eta}, \boldsymbol{\eta}^k),$$

  where $\bar{\ell} : \Theta \times \Theta \to \mathbb{R}$ satisfies:
  1. $\bar{\ell}(\boldsymbol{\eta}, \boldsymbol{\eta}) = \ell(\boldsymbol{\eta})$ for $\boldsymbol{\eta} \in \Theta$,
  2. and $\ell(\boldsymbol{\eta}) \leq \bar{\ell}(\boldsymbol{\eta}, \boldsymbol{\eta}')$ for $\boldsymbol{\eta}, \boldsymbol{\eta}' \in \Theta$.

- Adopting the simple linear majorization to make the convex function $h(\boldsymbol{\theta})$ affine by performing first-order Taylor expansion.
- Consequently, $\bar{l}(\boldsymbol{\theta}, \boldsymbol{\theta}^k) \triangleq g(\boldsymbol{\theta}) - h(\boldsymbol{\theta}^k) - \nabla_{\boldsymbol{\theta}}^T h(\boldsymbol{\theta}^k)(\boldsymbol{\theta} - \boldsymbol{\theta}^k)$.

- Adopting the simple linear majorization to make the convex function $h(\boldsymbol{\theta})$ affine by performing first-order Taylor expansion.
- Consequently, $\bar{l}(\boldsymbol{\theta}, \boldsymbol{\theta}^k) \triangleq g(\boldsymbol{\theta}) - h(\boldsymbol{\theta}^k) - \nabla_{\boldsymbol{\theta}}^T h(\boldsymbol{\theta}^k)(\boldsymbol{\theta} - \boldsymbol{\theta}^k)$.
- At each iteration, minimizing the MM cost function becomes a convex optimization problem.
- The converted problem can be further cast into a second-order cone program (SOCP) problem solved efficiently by MOSEK.

- Adopting the simple linear majorization to make the convex function $h(\boldsymbol{\theta})$ affine by performing first-order Taylor expansion.
- Consequently, $\bar{l}(\boldsymbol{\theta}, \boldsymbol{\theta}^k) \triangleq g(\boldsymbol{\theta}) - h(\boldsymbol{\theta}^k) - \nabla_{\boldsymbol{\theta}}^T h(\boldsymbol{\theta}^k)(\boldsymbol{\theta} - \boldsymbol{\theta}^k)$.
- At each iteration, minimizing the MM cost function becomes a convex optimization problem.
- The converted problem can be further cast into a second-order cone program (SOCP) problem solved efficiently by MOSEK.
- We also developed an ADMM algorithm that has the same computational complexity.

Rationale behind Sparsity Awareness

- Treat the overall GP model as a sum of Q independent GP models.
- Each subkernel $k_i(\boldsymbol{x}, \boldsymbol{x}') = \phi_i^T(\boldsymbol{x})\phi_i(\boldsymbol{x}')$, where $\phi_i(\boldsymbol{x}) : \mathbb{R}^L \mapsto \mathbb{R}^{L'}$, with $L' \gg L$.
- Then, $f(\boldsymbol{x}) = \sum_{i=1}^{Q} \boldsymbol{\theta}_i^T \phi_i(\boldsymbol{x})$, where $\boldsymbol{\theta}_i \sim \mathcal{N}(\mathbf{0}, \alpha_i \boldsymbol{I})$.

[7] D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," IEEE Trans. Signal Process., vol. 52, no. 8, pp. 2153-2164, Aug. 2004.

Rationale behind Sparsity Awareness

- Treat the overall GP model as a sum of Q independent GP models.
- Each subkernel $k_i(\boldsymbol{x}, \boldsymbol{x}') = \phi_i^T(\boldsymbol{x})\phi_i(\boldsymbol{x}')$, where $\phi_i(\boldsymbol{x}) : \mathbb{R}^L \mapsto \mathbb{R}^{L'}$, with $L' \gg L$.
- Then, $f(\boldsymbol{x}) = \sum_{i=1}^{Q} \boldsymbol{\theta}_i^T \phi_i(\boldsymbol{x})$, where $\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{0}, \alpha_i \boldsymbol{I})$.
- Mathematical proof follows the sparsity-promoting property of the *relevance vector machine* for classic sparse linear model. [7]

[7] D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," IEEE Trans. Signal Process., vol. 52, no. 8, pp. 2153-2164, Aug. 2004.

Rationale behind Sparsity Awareness

- Treat the overall GP model as a sum of Q independent GP models.
- Each subkernel $k_i(\boldsymbol{x}, \boldsymbol{x}') = \phi_i^T(\boldsymbol{x})\phi_i(\boldsymbol{x}')$, where $\phi_i(\boldsymbol{x}) : \mathbb{R}^L \mapsto \mathbb{R}^{L'}$, with $L' \gg L$.
- Then, $f(\boldsymbol{x}) = \sum_{i=1}^{Q} \boldsymbol{\theta}_i^T \phi_i(\boldsymbol{x})$, where $\boldsymbol{\theta}_i \sim \mathcal{N}(\mathbf{0}, \alpha_i \boldsymbol{I})$.
- Mathematical proof follows the sparsity-promoting property of the *relevance vector machine* for classic sparse linear model. [7]
- Maximizing the evidence is equivalent to finding the most relevant basis vectors in the over-complete dictionary, $\{\phi_i\}_{i=1}^{Q}$.

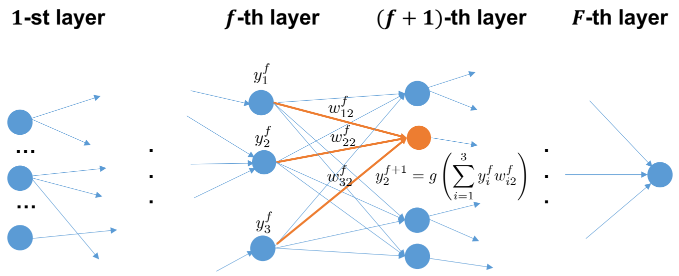
[7] D. P. Wipf and B. D. Rao, "Sparse Bayesian learning for basis selection," IEEE Trans. Signal Process., vol. 52, no. 8, pp. 2153-2164, Aug. 2004.

Outline

Section Goals

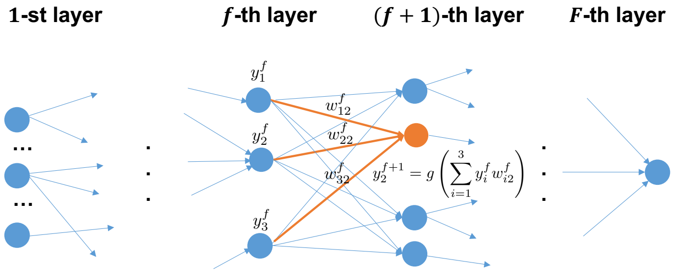
- We introduce sparsity-promoting techniques for pruning Bayesian deep neural networks (DNNs).
- That is, starting from a neural network with enormous number of nodes, to optimally remove nodes and/or links.
- We are going to follow two paths:
  1. **Path 1**: The parametric one via the Gaussian Scale Mixture (GSM) priors;
  2. **Path 2**: The non-parametric one via the Indian Buffet Process (IBP) prior.

Revisit Fundamentals of DNN

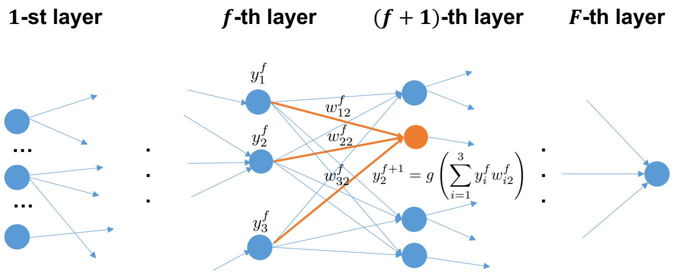


- Consider a fully connected DNN consisting of F layers.

Revisit Fundamentals of DNN

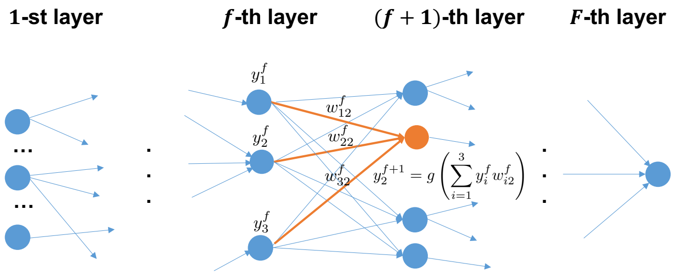


- Consider a fully connected DNN consisting of F layers.
- The number of nodes in the f-th $(1 \le f \le F-1)$ layer is a^f.
- For the i-th node in the f-th layer and the j-th node in the $(f+1)$-th layer, the link between them has a weight w_{ij}^f.



- The input vector to the $(f+1)$-th layer: $\boldsymbol{y}^f = [y_1^f, y_2^f, \cdots, y_{a^f}^f]^T$.
- The link weights associated with the j-th node:

$$\boldsymbol{w}_j^f = [w_{1j}^f, w_{2j}^f, \cdots, w_{a^f j}^f]^T.$$



- The input vector to the $(f+1)$-th layer: $\boldsymbol{y}^f = [y_1^f, y_2^f, \cdots, y_{a^f}^f]^T$.
- The link weights associated with the j-th node:

$$\boldsymbol{w}_j^f = [w_{1j}^f, w_{2j}^f, \cdots, w_{a^f j}^f]^T.$$

- The output of the j-th node is:

$$y_j^{f+1} = g\left(\sum_{i=1}^{a^f} w_{ij}^f y_i^f\right) = g\left(\left[\boldsymbol{w}_j^f\right]^T \boldsymbol{y}^f\right),$$

where $g(\cdot)$ is a nonlinear activation function.

Sparsity-Aware Modeling Using GSM Priors

- The idea dates back to the pioneering work of D. J. MacKay. [8]
- NN with a single hidden layer, the link weights can be treated as *random variables*.

[8] D. J. MacKay, "Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks," Network: Computation in Neural Systems, vol. 6, no. 3, pp. 469-505, 1995.

Sparsity-Aware Modeling Using GSM Priors

- The idea dates back to the pioneering work of D. J. MacKay. [8]
- NN with a single hidden layer, the link weights can be treated as *random variables*.
- The link weights can be modelled by zero-mean Gaussian priors, typically with a shared variance hyperparameter to encode the tendency of being zero/sparse.
- This induces an "inductive bias" of being sparse to the network.

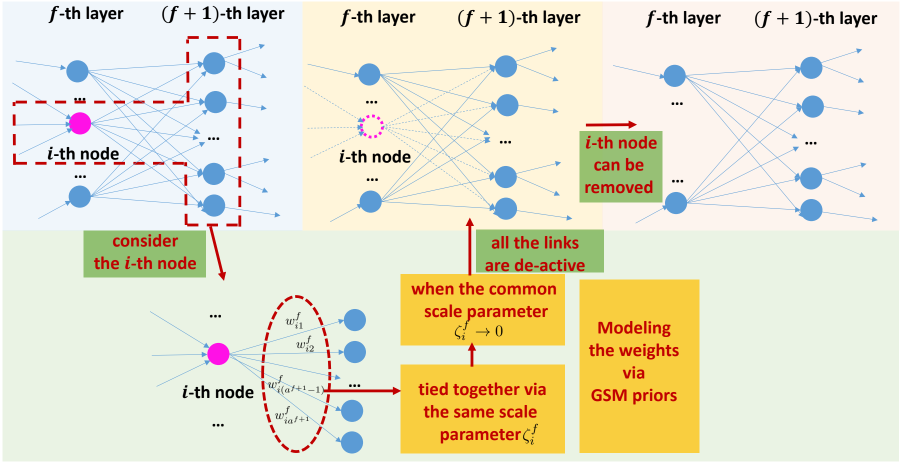
[8] D. J. MacKay, "Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks," Network: Computation in Neural Systems, vol. 6, no. 3, pp. 469-505, 1995.

Sparsity-Aware Modeling Using GSM Priors

- The idea dates back to the pioneering work of D. J. MacKay. [8]
- NN with a single hidden layer, the link weights can be treated as *random variables*.
- The link weights can be modelled by zero-mean Gaussian priors, typically with a shared variance hyperparameter to encode the tendency of being zero/sparse.
- This induces an "inductive bias" of being sparse to the network.
- The major difference between recent works and early ones lies in the adopted priors.

[8] D. J. MacKay, "Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks," Network: Computation in Neural Systems, vol. 6, no. 3, pp. 469-505, 1995.

- Global picture of node-wise sparsity rationale.



- For each random weight w_{ij}^f, we adopt a sparsity-promoting GSM prior:

$$p(w_{ij}^f) = \int \mathcal{N}(w_{ij}^f; 0, \zeta_{ij}^f) p(\zeta_{ij}^f; \boldsymbol{\eta}) d\zeta_{ij}^f.$$

- For each random weight w_{ij}^{f}, we adopt a sparsity-promoting GSM prior:

$$p(w_{ij}^{f}) = \int \mathcal{N}(w_{ij}^{f}; 0, \zeta_{ij}^{f}) p(\zeta_{ij}^{f}; \boldsymbol{\eta}) d\zeta_{ij}^{f}.$$

- Functional form of $p(\zeta_{ij}^{f}; \boldsymbol{\eta})$ below corresponds to a GSM prior.

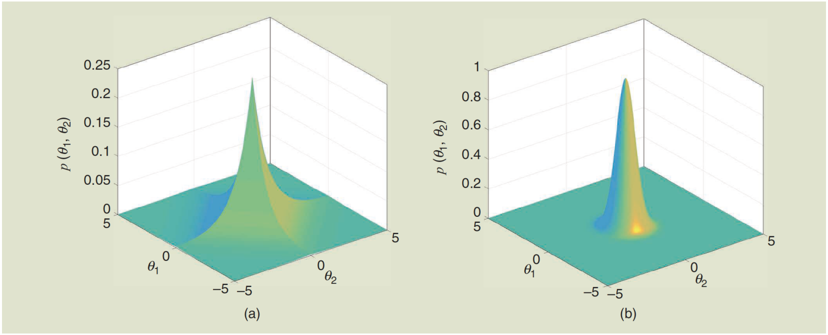
GSM prior $p(w_l)$	Mixing distribution $p(\zeta_l)$
Student's t	Inverse Gamma: $p(\zeta_l; \boldsymbol{\eta}_p = [a, b]) = \mathrm{IG}(\zeta_l; a, b)$
Normal-Jefferys	Log-uniform: $p(\zeta_l; \boldsymbol{\eta}_p = [\,]) \propto \frac{1}{\lvert\zeta_l\rvert}$
Laplacian	Gamma: $p(\zeta_l; \boldsymbol{\eta}_p = [a, b]) = \mathrm{Ga}(\zeta_l; a, b)$
Generalized hyperbolic	Generalized inverse Gaussian: $p(\zeta_l; \boldsymbol{\eta}_p = [a, b, \lambda]) = \mathrm{GIG}(\zeta_l; a, b, \lambda)$
Horseshoe	$\zeta_l = \tau_l \upsilon_l, \boldsymbol{\eta}_p = [a, b]$ Half Cauchy: $p(\tau_l) = C^{+}(0, a)$ $p(\upsilon_l) = C^{+}(0, b)$

- Next, we show how to conduct node-wise sparsity-aware modeling.
- Group the weights $\{w_{ij}^f\}_{j=1}^{a^{f+1}}$ connected to the i-th node, and assign a common scale parameter ζ_i^f to their GSM priors, i.e., $\zeta_{ij}^f = \zeta_i^f, \forall j$.

- Next, we show how to conduct node-wise sparsity-aware modeling.
- Group the weights $\{w_{ij}^{f}\}_{j=1}^{a^{f+1}}$ connected to the i-th node, and assign a common scale parameter ζ_i^f to their GSM priors, i.e., $\zeta_{ij}^f = \zeta_i^f, \forall j$.
- The prior modeling for the i-th node related weights $\{w_{ij}^{f}\}_{j=1}^{a^{f+1}}$:

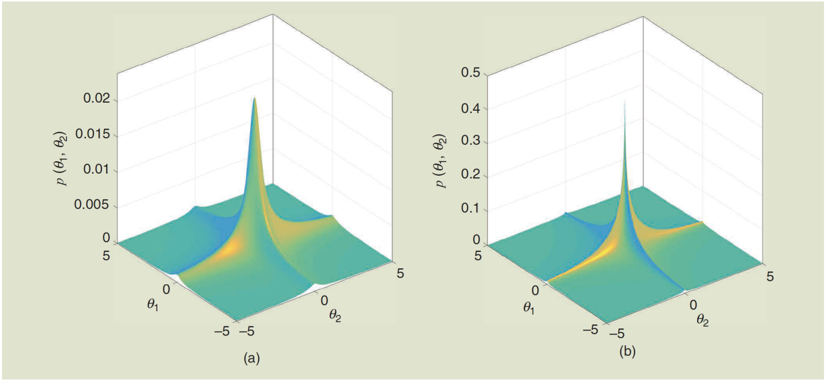
$$p(\{w_{ij}^{f}\}_{j=1}^{a^{f+1}}) = \int p(\{w_{ij}^{f}\}_{j=1}^{a^{f+1}} | \zeta_i^f) p(\zeta_i^f; \boldsymbol{\eta}) d\zeta_i^f$$

$$= \int \prod_{j=1}^{a^{f+1}} \mathcal{N}(w_{ij}^f; 0, \zeta_i^f) p(\zeta_i^f; \boldsymbol{\eta}) d\zeta_i^f.$$



The joint probability distribution of the model parameters in 2D space. (a) The Laplacian distribution. (b) The Gaussian distribution.

- Heavy-tail Laplacian distribution peaks sharply around zero and falls slowly along the axes, thus promoting sparse solutions.
- Gaussian distribution decays more rapidly along both dimensions when compared to the Laplacian distribution.



(a) The Student's t distribution versus (b) the horseshoe distribution.

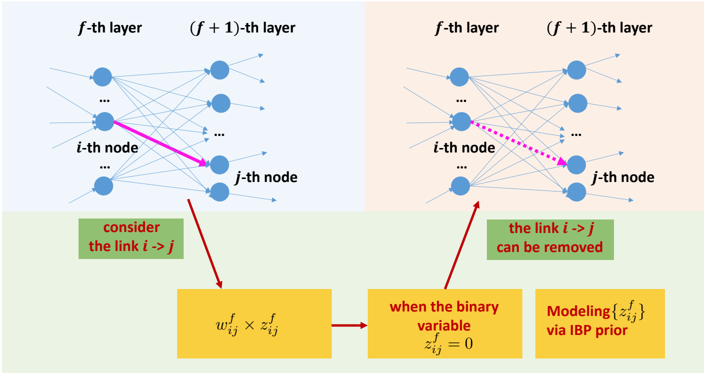
Sparsity-Aware Modeling Using IBP Prior

- The previous approach has a major drawback, i.e., the number of nodes per layer has to be specified and pre-selected.

Sparsity-Aware Modeling Using IBP Prior

- The previous approach has a major drawback, i.e., the number of nodes per layer has to be specified and pre-selected.
- In contrast, we now turn our attention to non-parametric techniques for link-wise sparsity.
- We are going to assume that the nodes per layer are theoretically infinite (in practice a large enough number) and then use the IBP prior to enforce sparsity.

- Global picture of the rationale behind link-wise sparsity-aware modeling.



f-th layer
$(f+1)$-th layer
...
i-th node
j-th node
f-th layer
$(f+1)$-th layer
i-th node
j-th node
consider the link i -> j
the link i -> j can be removed
$w_{ij}^f \times z_{ij}^f$
when the binary variable $z_{ij}^f = 0$
Modeling $\{z_{ij}^f\}$ via IBP prior

- We multiply each weight, i.e., w_{ij}^f, with a corresponding auxiliary (hidden) binary random variable, z_{ij}^f.

Brief Introduction to IBP Prior

- How to adapt IBP prior to fit our needs for designing BNNs?
- Assuming an Indian restaurant offers $K \to \infty$ of dishes (output dimension) and there are L customers (input dimension).

Brief Introduction to IBP Prior

- How to adapt IBP prior to fit our needs for designing BNNs?
- Assuming an Indian restaurant offers $K \to \infty$ of dishes (output dimension) and there are L customers (input dimension).
- Rationale of IBP:
  1. The first dimension, say, x_1 is linked to some of the infinite nodes with certain probabilities;
  2. The second dimension, x_2 is linked to some of the previously linked nodes and to some new ones according to certain probabilities;
  3. ... until x_L.

Brief Introduction to IBP Prior

- How to adapt IBP prior to fit our needs for designing BNNs?
- Assuming an Indian restaurant offers $K \to \infty$ of dishes (output dimension) and there are L customers (input dimension).
- Rationale of IBP:
  1. The first dimension, say, x_1 is linked to some of the infinite nodes with certain probabilities;
  2. The second dimension, x_2 is linked to some of the previously linked nodes and to some new ones according to certain probabilities;
  3. ... until x_L.
- Mathematically, we aim to generate a series of binary random variables, $z_{ij} \in \{0, 1\}$, $i = 1, 2, \ldots, L$ and $j = 1, 2, \ldots$.
- If $z_{ij} = 1$, the i-th customer (i-th dimension) selects the j-th dish (linked to the j-th node).

- Following the stick-breaking construction idea, first generate:

$$u_j \sim \text{Beta}(u_j|\alpha, 1), \quad \pi_j = \prod_{l=1}^{j} u_l, \; j = 1, 2, \cdots .$$

- Then, the generated probabilities, π_j, are used to populate the matrix $\boldsymbol{Z}$, by drawing samples from a Bernoulli distribution:

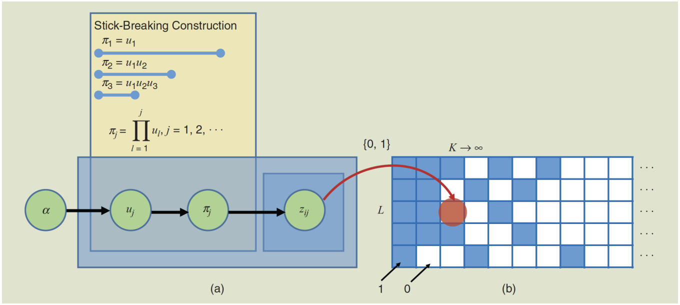
$$z_{ij} \sim \text{Bernoulli}(z_{ij}|\pi_j), \quad i = 1, 2, \cdots , L.$$

- Following the stick-breaking construction idea, first generate:

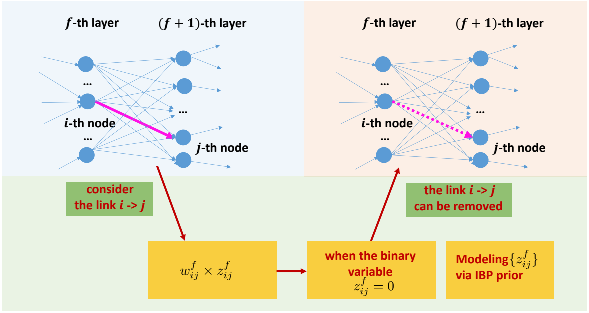
$$u_j \sim \text{Beta}(u_j|\alpha, 1), \quad \pi_j = \prod_{l=1}^{j} u_l, \; j = 1, 2, \cdots .$$

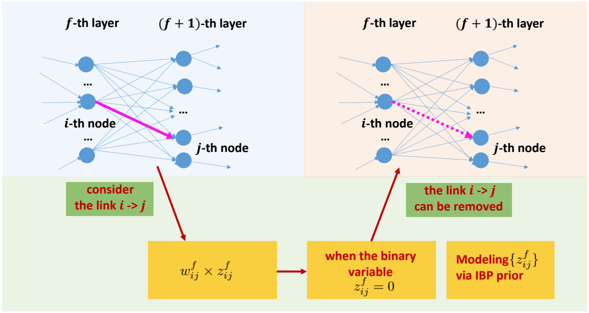
- Then, the generated probabilities, π_j, are used to populate the matrix $\boldsymbol{Z}$, by drawing samples from a Bernoulli distribution:

$$z_{ij} \sim \text{Bernoulli}(z_{ij}|\pi_j), \quad i = 1, 2, \cdots , L.$$



(a):Beta-Bernoulli model. (b): The binary matrix $\boldsymbol{Z}$.


f-th layer
$(f+1)$-th layer
i-th node
j-th node
f-th layer
$(f+1)$-th layer
i-th node
j-th node
consider the link i -> j
the link i -> j can be removed
$w_{ij}^f \times z_{ij}^f$
when the binary variable $z_{ij}^f = 0$
Modeling $\{z_{ij}^f\}$ via IBP prior


f-th layer
$(f+1)$-th layer
i-th node
j-th node
f-th layer
$(f+1)$-th layer
i-th node
j-th node
consider the link i -> j
the link i -> j can be removed
$w_{ij}^f \times z_{ij}^f$
when the binary variable $z_{ij}^f = 0$
Modeling $\{z_{ij}^f\}$ via IBP prior

- $\{\{z_{ij}^f\}_{i=1}^{a^f}\}_{j=1}^{a^{f+1}}$ are generated via the stick-breaking IBP prior:

$$u_j^f \sim \mathrm{Beta}(u_j^f|\alpha, 1), \quad \pi_j^f = \prod_{l=1}^{j} u_l^f, \quad z_{ij}^f \sim \mathrm{Bernoulli}(z_{ij}^f|\pi_j^f).$$

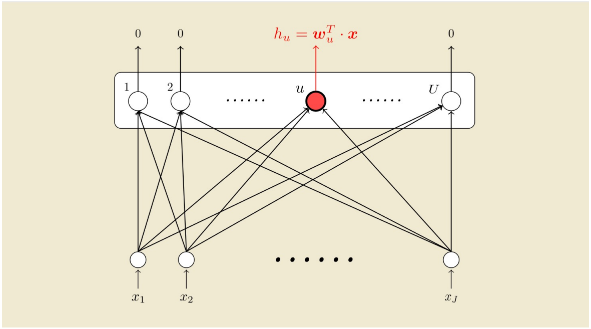
- Binary matrix $\boldsymbol{Z}^f \in \mathbb{R}^{a^f \times a^{f+1}}$, with its (ij)-th element being z_{ij}^f for the f-th layer.

- The flexibility of the link-wise formulation allows us to go one step further.
- Recently, the stick-breaking IBP prior was combined with a radically different, biologically-inspired and competition-based activation, namely the stochastic local winner-takes-all (LWTA). [9],[10], [11]
- LWTA adopts blocks of competing linear units/neurons against ordinary neurons with nonlinear activation for nonlinearity.

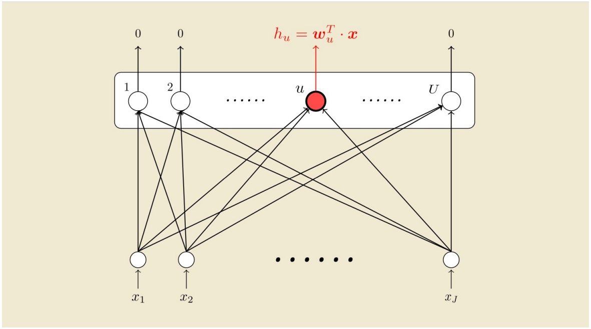
[9] K. Panousis, S. Chatzis, and S. Theodoridis, “Nonparametric Bayesian deep networks with local competition,” in ICML, 2019.

[10] K. Panousis, S. Chatzis, A. Alexos, and S. Theodoridis, “Local competition and stochasticity for adversarial robustness in deep learning,” in AISTAT, 2021.

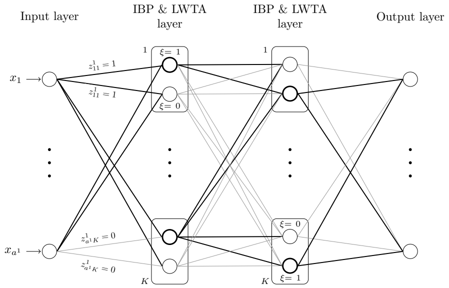
[11] K. Panousis, S. Chatzis, and S. Theodoridis, “Stochastic local winner-takes-all networks enable profound adversarial robustness,” in NeurIPS, 2021.



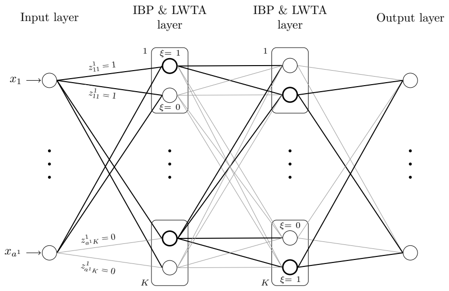
- Each node comprises a set of linear (inner product) units.



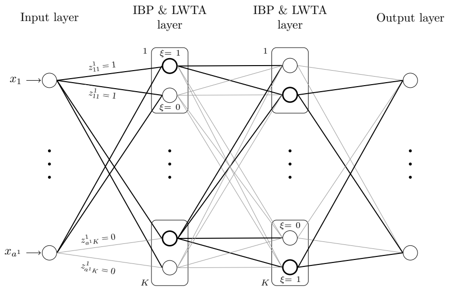
- Each node comprises a set of linear (inner product) units.
- The unit with the *strongest* activation is deemed to be the *winner* and passes its output to the next layer, while the rest are being zero.
- This deterministic winner selection is known as *hard* LWTA.
- We turn to stochastic LWTA with enhanced performance.



- L inputs, K LWTA blocks; J linear units.
- Here, $L = a^1$, $J = 2$ for each layer.



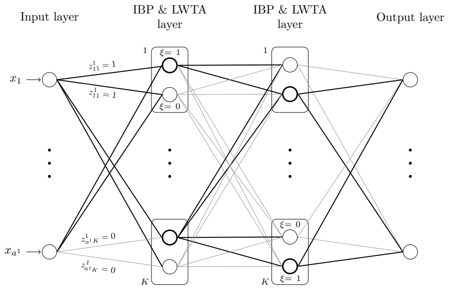
- L inputs, K LWTA blocks; J linear units.
- Here, $L = a^1$, $J = 2$ for each layer.
- z_{ik} is modeled by IBP as introduced before.
- Weights: w_{ikj}, $i = 1, 2, \ldots L$, $k = 1, 2, \ldots, K$, $j = 1, 2 \ldots, J$.



- L inputs, K LWTA blocks; J linear units.
- Here, $L = a^1$, $J = 2$ for each layer.
- z_{ik} is modeled by IBP as introduced before.
- Weights: $w_{ikj},\ i = 1, 2, \ldots L,\ k = 1, 2, \ldots, K,\ j = 1, 2 \ldots, J$.

Linear unit output is:

$$y_{kj} = \xi_{kj} \boldsymbol{w}_{kj}^T \boldsymbol{x} = \xi_{kj} \sum_{i=1}^{L} w_{ikj} x_i, \quad \xi_{kj} \in \{0, 1\}, \ \sum_{j=1}^{J} \xi_{kj} = 1.$$



- The output depends on the value of $\xi_{kj} \in \{0, 1\}$.
- The respective probabilities, which control the value of ξ_{kj}, are computed via *softmax*:

$$P_{kj} = \frac{\exp(h_{kj})}{\sum_{j=1}^{J} \exp(h_{kj})},$$

$$h_{kj} = \sum_{i=1}^{L} (z_{ik} w_{ikj}) x_i.$$

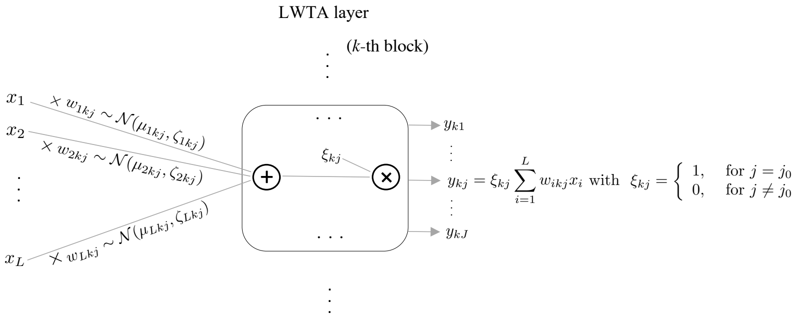
Inference Algorithms for Bayesian DNNs

- Inference (training) for Bayesian deep neural networks follows the same backpropagation-type of philosophy as that of deterministic DNNs.

Inference Algorithms for Bayesian DNNs

- Inference (training) for Bayesian deep neural networks follows the same backpropagation-type of philosophy as that of deterministic DNNs.
- There are, however, two notable differences:
  - First, the unknown (synaptic) parameters/weights are now described via parameterized distributions.
  - Second, the evidence function to be maximized is intractable and has to be approximated by its ELBO.

- We focus on the Bayesian deep network that comprises layers with:
  - stochastic LWTA blocks;
  - stochastic synaptic weights of Gaussian form;
  - a sparsity-inducing mechanism imposed over the network synapses that is driven via an IBP prior.


LWTA layer
(k-th block)
x_1
x_2
x_L
$\times\, w_{1kj} \sim \mathcal{N}(\mu_{1kj}, \zeta_{1kj})$
$\times\, w_{2kj} \sim \mathcal{N}(\mu_{2kj}, \zeta_{2kj})$
$\times\, w_{Lkj} \sim \mathcal{N}(\mu_{Lkj}, \zeta_{Lkj})$
ξ_{kj}
y_{k1}
y_{kJ}

$$y_{kj} = \xi_{kj} \sum_{i=1}^{L} w_{ikj} x_i \text{ with } \xi_{kj} = \begin{cases} 1, & \text{for } j = j_0 \\ 0, & \text{for } j \neq j_0 \end{cases}$$

- Synaptic weights:

Prior : $p(w_{ikj}) \sim \mathcal{N}(w_{ikj}|0,1)$, Posterior : $q(w_{ikj}) \sim \mathcal{N}(w_{ikj}|\mu_{ikj}, \zeta_{ikj})$

- Utility binary random variables:

Prior : $\text{Beta}(u_k|\alpha, 1)$, Posterior : $q(u_k) = \text{Beta}(u_k|a_k, b_k)$

Prior : $\text{Bernoulli}(z_{ik}|\pi_{ik})$, Posterior : $q(z_{ik}) = \text{Bernoulli}(z_{ik}|\tilde{\pi}_{ik})$

- Indicator random vectors, $\boldsymbol{\xi}_k$:

Prior : $p(\boldsymbol{\xi}_k) = \text{Categorical}(\boldsymbol{\xi}_k|\frac{1}{J}, \ldots, \frac{1}{J})$, i.e., all linear units equiprobable.

Posterior : $q(\boldsymbol{\xi}_k) = \text{Categorical}\left(\boldsymbol{\xi}_k \middle| P_{k1}, \ldots, P_{kJ}\right)$

- Synaptic weights:

$$\text{Prior}: p(w_{ikj}) \sim \mathcal{N}(w_{ikj}|0,1), \quad \text{Posterior}: \ q(w_{ikj}) \sim \mathcal{N}(w_{ikj}|\mu_{ikj}, \zeta_{ikj})$$

- Utility binary random variables:

$$\text{Prior}: \text{Beta}(u_k|\alpha, 1), \quad \text{Posterior}: \ q(u_k) = \text{Beta}(u_k|a_k, b_k)$$

$$\text{Prior}: \text{Bernoulli}(z_{ik}|\pi_{ik}), \quad \text{Posterior}: \ q(z_{ik}) = \text{Bernoulli}(z_{ik}|\tilde{\pi}_{ik})$$

- Indicator random vectors, $\boldsymbol{\xi}_k$:

$$\text{Prior}: \ p(\boldsymbol{\xi}_k) = \text{Categorical}(\boldsymbol{\xi}_k|\frac{1}{J}, \ldots, \frac{1}{J}), \text{ i.e., all linear units equiprobable.}$$

$$\text{Posterior}: \ q(\boldsymbol{\xi}_k) = \text{Categorical}\left(\boldsymbol{\xi}_k \middle| P_{k1}, \ldots, P_{kJ}\right)$$

- The hyperparameters to be optimized include:

$$\boldsymbol{\theta} = [\{\mu_{ikj}\}, \{\zeta_{ikj}\}, \{a_k\}, \{b_k\}, \{\tilde{\pi}_{ik}\}].$$

- $\mathcal{D}$: input-output training dataset;
- $\mathbf{Z}$: set of the synaptic utility indicators across the network layers;
- $\mathbf{\Xi}$: set of winner unit indicators across all blocks of all layers;
- $\mathbf{W}$: set of synapse weights across all layers;
- $\mathbf{U}$: set of the stick-variables of the sparsity-inducing priors imposed across the network layers.

- $\mathcal{D}$: input-output training dataset;
- $\mathbf{Z}$: set of the synaptic utility indicators across the network layers;
- $\mathbf{\Xi}$: set of winner unit indicators across all blocks of all layers;
- $\mathbf{W}$: set of synapse weights across all layers;
- $\mathbf{U}$: set of the stick-variables of the sparsity-inducing priors imposed across the network layers.

- Employing the mean-field approximation $q(\mathbf{W}, \mathbf{Z}, \mathbf{\Xi}, \mathbf{U})$ to obtain:

$$\begin{aligned}\text{ELBO}(\boldsymbol{\theta}) =& \mathbb{E}_q\Big[\sum_{n=1}^{N}\sum_{c=1}^{C} y_{nc} \ln \tilde{y}_{nc}(\boldsymbol{x}_n; \mathbf{W}, \mathbf{Z}, \mathbf{\Xi}, \mathbf{U})\Big] \\ & + \underbrace{\mathbb{E}_q \ln \frac{p(\mathbf{Z}|\mathbf{U})}{q(\mathbf{Z})} + \mathbb{E}_q \ln \frac{p(\mathbf{U})}{q(\mathbf{U})}}_{\text{regularizing terms}} + \underbrace{\mathbb{E}_q \ln \frac{p(\mathbf{\Xi})}{q(\mathbf{\Xi}|\mathbf{Z}, \mathbf{W})} + \mathbb{E}_q \ln \frac{p(\mathbf{W})}{q(\mathbf{W})}}_{\text{regularizing terms}}\end{aligned}$$

- To approximate the expectations in ELBO via samples, we introduce reparameterization trick.
  Example: Let $\mathcal{N}(w_{ikj}|\mu_{ikj}, \zeta_{ikj})$ be the current estimate. We have

$$\tilde{w}_{ikj} = \mu_{ikj} + \zeta_{ikj}^{1/2} \cdot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 1).$$

- In this way, every link is determined explicitly by the pair (μ_{ijk}, ζ_{ijk}); and the backpropagation optimizes the means and variances.
- Reparameterization of the rest variables follows a similar rationale.

[9] K. Panousis, S. Chatzis, and S. Theodoridis, "Nonparametric Bayesian deep networks with local competition," in ICML, 2019.

- To approximate the expectations in ELBO via samples, we introduce reparameterization trick.
  Example: Let $\mathcal{N}(w_{ikj}|\mu_{ikj}, \zeta_{ikj})$ be the current estimate. We have

$$\tilde{w}_{ikj} = \mu_{ikj} + \zeta_{ikj}^{1/2} \cdot \epsilon, \qquad \epsilon \sim \mathcal{N}(0,1).$$

- In this way, every link is determined explicitly by the pair (μ_{ijk}, ζ_{ijk}); and the backpropagation optimizes the means and variances.
- Reparameterization of the rest variables follows a similar rationale.
- Off-the-shelf gradient-based optimizer, e.g., the Adam, can be used for training.
- More details about the inference algorithm can be found here.[9]

[9] K. Panousis, S. Chatzis, and S. Theodoridis, "Nonparametric Bayesian deep networks with local competition," in ICML, 2019.

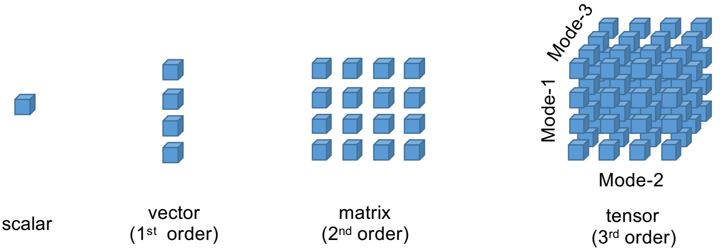
Outline

Section Goals

- So far, we have elucidated the sparsity-aware learning for the two recent supervised data analysis tools, namely the DNNs and GPs.
- Their underlying idea has inspired recent study for the unsupervised learning tools, e.g., tensor decomposition.
- We take the most fundamental *canonical polyadic decomposition* (CPD) as an example to elucidate the key ideas of
  - Prior Modeling;
  - Inference Method.

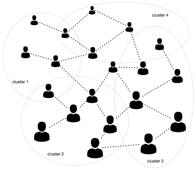
Tensors and CPD

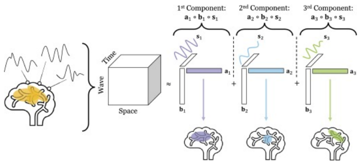
- Tensors are regarded as multi-dimensional generalization of matrices.



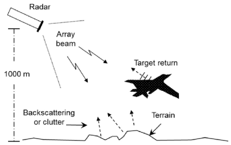
- Specifically, a P-dimensional (P-D) dataset can be represented by a P-D tensor $\mathcal{D} \in \mathbb{R}^{J_1 \times J_2 \times \cdots J_P}$.

- Given a tensor $\mathcal{D}$, unsupervised learning aims to identify the underlying source signals that generated the observed data:
  - "Clustering" in social network analysis
  - "Blind source separation" in EEG data analysis
  - "Blind signal estimation" in radar/sonar signal processing





Time
Wave
Space
≈
1st Component: $\mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{s}_1$
2nd Component: $\mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{s}_2$
3rd Component: $\mathbf{a}_3 \circ \mathbf{b}_3 \circ \mathbf{s}_3$
$\mathbf{s}_1$
$\mathbf{s}_2$
$\mathbf{s}_3$
$\mathbf{a}_1$
$\mathbf{a}_2$
$\mathbf{a}_3$
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{b}_3$
+
+



Figures are from [12]-[13].

[12] S. Redif, S. Weiss, and J. G. McWhirter. Relevance of polynomial matrix decompositions to broadband blind signal separation. Signal processing, 134, 76-86, 2017.

[13] https://www.sciencedirect.com/topics/engineering/blind-signal-separation

Tensor CPD has been proven to be a powerful tool with good interpretability. It is formally defined as follows[14].

Definition of Tensor CPD

Given a P-D tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots J_P}$, CPD seeks to find the vectors $\{\boldsymbol{a}_r^{(1)}, \boldsymbol{a}_r^{(2)}, \cdots, \boldsymbol{a}_r^{(P)}\}_{r=1}^{R}$ such that

$$\begin{aligned}\boldsymbol{\mathcal{D}} &= \sum_{r=1}^{R} \underbrace{\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(P)}}_{\text{rank-1 tensor}}, \\ &\triangleq [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(P)}]\!],\end{aligned}$$

where $\circ$ denotes vector outer product; $\boldsymbol{A}^{(p)} \triangleq [\boldsymbol{a}_1^{(p)}, \boldsymbol{a}_2^{(p)}, \cdots, \boldsymbol{a}_R^{(p)}] \in \mathbb{R}^{J_p \times R}, \forall p$, is called the factor matrix. The minimal number R that yields the above expression is termed as the tensor rank.

[14] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis and C. Faloutsos, "Tensor Decomposition for Signal Processing and Machine Learning," in IEEE Transactions on Signal Processing, vol. 65, no. 13, pp. 3551-3582, 2017.

Inspect the definition of tensor CPD:

$$\mathcal{D} = \sum_{r=1}^{R} \underbrace{\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(P)}}_{\text{rank-1 tensor}},$$
$$\triangleq [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(P)}]\!],$$

- When $P = 2$, it reduces to decomposing a matrix $\boldsymbol{D} \in \mathbb{R}^{J_1 \times J_2}$ into the summation of R rank-1 matrices, i.e., $\boldsymbol{D} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)}$.

Inspect the definition of tensor CPD:

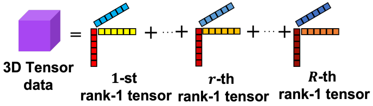
$$\boldsymbol{\mathcal{D}} = \sum_{r=1}^{R} \underbrace{\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(P)}}_{\text{rank-1 tensor}},$$
$$\triangleq [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(P)}]\!],$$

- When $P = 2$, it reduces to decomposing a matrix $\boldsymbol{D} \in \mathbb{R}^{J_1 \times J_2}$ into the summation of R rank-1 matrices, i.e., $\boldsymbol{D} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)}$.
- By defining the term $\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(P)}$ as a P-D rank-1 tensor, CPD essentially seeks for R rank-1 tensors/components from the observed dataset.

Inspect the definition of tensor CPD:

$$\boldsymbol{\mathcal{D}} = \sum_{r=1}^{R} \underbrace{\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(P)}}_{\text{rank-1 tensor}},$$
$$\triangleq [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(P)}]\!],$$

- When $P = 2$, it reduces to decomposing a matrix $\boldsymbol{D} \in \mathbb{R}^{J_1 \times J_2}$ into the summation of R rank-1 matrices, i.e., $\boldsymbol{D} = \sum_{r=1}^{R} \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)}$.
- By defining the term $\boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(P)}$ as a P-D rank-1 tensor, CPD essentially seeks for R rank-1 tensors/components from the observed dataset.
- Each rank-1 tensor corresponds to one specific underlying source signal.
- The tensor rank R has a clear physical meaning as the number of the underlying source signals.



Low-Rank CPD and Sparsity-Aware Modeling

- In real-world applications, the number of source signals is usually small.
- For instance, in brain-source imaging, the EEG data analysis has shown that only a small fraction of source signals contribute to the brain activities.

[15] L. Cheng, Z. Chen, Y. -C. Wu and S. Theodoridis, "Towards Flexible Sparsity-Aware Modeling: Automatic Tensor Rank Learning Using the Generalized Hyperbolic Prior," IEEE Transactions on Signal Processing, vol. 70, pp. 1834-1849, 2022.

Low-Rank CPD and Sparsity-Aware Modeling

- In real-world applications, the number of source signals is usually small.
- For instance, in brain-source imaging, the EEG data analysis has shown that only a small fraction of source signals contribute to the brain activities.
- This suggests that the CPD model should assume a low tensor rank R to avoid data overfitting.
- In the sequel, we show how the *low-rankness* is probabilistically embedded into the CPD model through practicing the ideas reported previously[15].

[15] L. Cheng, Z. Chen, Y. -C. Wu and S. Theodoridis, "Towards Flexible Sparsity-Aware Modeling: Automatic Tensor Rank Learning Using the Generalized Hyperbolic Prior," IEEE Transactions on Signal Processing, vol. 70, pp. 1834–1849, 2022.

- We employ an over-parameterized model for CPD by assuming an upper-bound value L of tensor rank R, i.e., $L \gg R$.
- The low-rankness implies that $(L-R)$ rank-1 tensors should be zero, each specified by vectors $\{\boldsymbol{a}_l^{(p)}\}_{p=1}^{P}, \forall l$.
- Let vector $\mathfrak{q}_l \triangleq [\boldsymbol{a}_l^{(1)}; \boldsymbol{a}_l^{(2)}; \cdots ; \boldsymbol{a}_l^{(P)}] \in \mathbb{R}^{\sum_{p=1}^{P} J_p}, \forall l$. The low-rankness indicates that a number of vectors in the set $\{\mathfrak{q}_l\}_{l=1}^{L}$ are zero vectors.

- We employ an over-parameterized model for CPD by assuming an upper-bound value L of tensor rank R, i.e., $L \gg R$.
- The low-rankness implies that $(L - R)$ rank-1 tensors should be zero, each specified by vectors $\{\boldsymbol{a}_l^{(p)}\}_{p=1}^{P}, \forall l$.
- Let vector $\mathfrak{q}_l \triangleq [\boldsymbol{a}_l^{(1)}; \boldsymbol{a}_l^{(2)}; \cdots ; \boldsymbol{a}_l^{(P)}] \in \mathbb{R}^{\sum_{p=1}^{P} J_p}, \forall l$. The low-rankness indicates that a number of vectors in the set $\{\mathfrak{q}_l\}_{l=1}^{L}$ are zero vectors.
- To model such sparsity, we adopt the GSM prior:

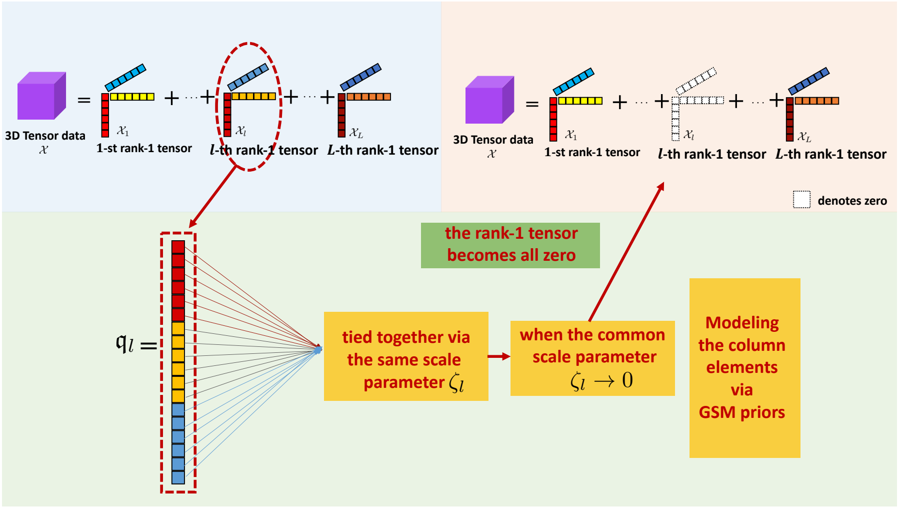
$$
\begin{aligned}
p(\mathfrak{q}_l) &= \int \prod_{i=1}^{\sum_{p=1}^{P} J_p} \mathcal{N}([\mathfrak{q}_l]_i; 0, \zeta_l) p(\zeta_l; \boldsymbol{\eta}_p) d\zeta_l, \\
&= \int \prod_{p=1}^{P} \mathcal{N}(\boldsymbol{a}_l^{(p)}; 0, \zeta_l \boldsymbol{I}) p(\zeta_l; \boldsymbol{\eta}_p) d\zeta_l.
\end{aligned}
$$

- To model such sparsity, we adopt the GSM prior:

$$p(\mathfrak{q}_l) = \int \prod_{i=1}^{\sum_{p=1}^{P} J_p} \mathcal{N}([\mathfrak{q}_l]_i; 0, \zeta_l) p(\zeta_l; \boldsymbol{\eta}_p) d\zeta_l,$$

$$= \int \prod_{p=1}^{P} \mathcal{N}(\boldsymbol{a}_l^{(p)}; 0, \zeta_l \boldsymbol{I}) p(\zeta_l; \boldsymbol{\eta}_p) d\zeta_l.$$

GSM prior $p(\mathfrak{q}_l)$	Mixing distribution $p(\zeta_l)$
Student's t	Inverse Gamma: $p(\zeta_l; \boldsymbol{\eta}_p = [a, b]) = \mathrm{IG}(\zeta_l; a, b)$
Normal-Jefferys	Log-uniform: $p(\zeta_l; \boldsymbol{\eta}_p = [\ ]) \propto \frac{1}{\lvert\zeta_l\rvert}$
Laplacian	Gamma: $p(\zeta_l; \boldsymbol{\eta}_p = [a, b]) = \mathrm{Ga}(\zeta_l; a, b)$
Generalized hyperbolic	Generalized inverse Gaussian: $p(\zeta_l; \boldsymbol{\eta}_p = [a, b, \lambda]) = \mathrm{GIG}(\zeta_l; a, b, \lambda)$
Horseshoe	$\zeta_l = \tau_l \upsilon_l, \boldsymbol{\eta}_p = [a, b]$ Half Cauchy: $p(\tau_l) = C^+(0, a)$ $p(\upsilon_l) = C^+(0, b)$



3D Tensor data $\mathcal{X}$
=
$\mathcal{X}_1$
1-st rank-1 tensor
+ ··· +
$\mathcal{X}_l$
l-th rank-1 tensor
+ ··· +
$\mathcal{X}_L$
L-th rank-1 tensor
3D Tensor data $\mathcal{X}$
=
$\mathcal{X}_1$
1-st rank-1 tensor
+ ··· +
$\mathcal{X}_l$
l-th rank-1 tensor
+ ··· +
$\mathcal{X}_L$
L-th rank-1 tensor
denotes zero
$\mathfrak{q}_l =$
the rank-1 tensor becomes all zero
tied together via the same scale parameter ζ_l
when the common scale parameter $\zeta_l \rightarrow 0$
Modeling the column elements via GSM priors

Remark 1 (Nonnegative Modeling)

If the factor matrices are further constrained to be non-negative for enhanced interpretability in certain applications, simple modification, that is, multiplying a unit-step function $\mathrm{U}(\boldsymbol{a}_l^{(p)} \geq 0)$ (which returns one when $\boldsymbol{a}_l^{(p)} \geq 0$ or zero otherwise) to the prior derived in the last slide, can be made to embed both the non-negativeness and the low-rankness[16].

[16] L. Cheng, X. Tong, S. Wang, Y. -C. Wu and H. V. Poor, "Learning Nonnegative Factors From Tensor Data: Probabilistic Modeling and Inference Algorithm," in IEEE Transactions on Signal Processing, vol. 68, pp. 1792-1806, 2020.

Remark 2 (Extensions to Other Tensor Models)

- Similar ideas have been applied to other tensor decomposition models including Tucker decomposition (TuckerD) and tensor train decomposition (TTD)[17].
- In these works, one first assumes an over-parameterized model by setting the model configuration parameters (e.g., multi-linear ranks in TuckerD and TT ranks in TTD) to be large numbers, and then imposes GSM prior on the associated model parameters to control the model complexity.

[17] L. Cheng, Z. Chen, and Y.-C. Wu. "Bayesian Tensor Decomposition for Signal Processing and Machine Learning", Springer, 2023.

Inference Algorithms for Bayesian TDs

- Now, we introduce the inference algorithm design for Bayesian tensor decompositions.
- Our focus is on the key ideas for inferring the Bayesian tensor CPD model with the Gaussian likelihood and the GSM prior.
- For other tensor decomposition formats, the associated inference algorithm follows a similar rationale[17].

[17] L. Cheng, Z. Chen, and Y.-C. Wu. "Bayesian Tensor Decomposition for Signal Processing and Machine Learning", Springer, 2023.

- The goal of inference is to estimate the posterior distributions of factor matrices $\{\boldsymbol{A}^{(p)} \in \mathbb{R}^{J_p \times L}\}_{p=1}^{P}$ from possibly incomplete P-D tensor data observations $\boldsymbol{\mathcal{Y}}_{\boldsymbol{\Omega}} \in \mathbb{R}^{J_1 \times \cdots \times J_P}$.
- $\boldsymbol{\mathcal{Y}}_{j_1,\cdots,j_P}$ is observed if the P-tuple indices $(j_1, \cdots, j_P)$ belongs to $\boldsymbol{\Omega}$.

- The goal of inference is to estimate the posterior distributions of factor matrices $\{\boldsymbol{A}^{(p)} \in \mathbb{R}^{J_p \times L}\}_{p=1}^{P}$ from possibly incomplete P-D tensor data observations $\boldsymbol{\mathcal{Y}}_{\boldsymbol{\Omega}} \in \mathbb{R}^{J_1 \times \cdots \times J_P}$.
- $\boldsymbol{\mathcal{Y}}_{j_1,\cdots,j_P}$ is observed if the P-tuple indices $(j_1, \cdots, j_P)$ belongs to $\boldsymbol{\Omega}$.
- The forward problem is commonly modeled as a Gaussian likelihood:

$$p(\boldsymbol{\mathcal{Y}}_{\boldsymbol{\Omega}} | \{\boldsymbol{A}^{(p)}\}_{p=1}^{P}; \beta) = \prod_{(j_1,\cdots,j_P)\in\boldsymbol{\Omega}} \mathcal{N}(\boldsymbol{\mathcal{Y}}_{j_1,\cdots,j_P} \; ; \; [\![\boldsymbol{A}^{(1)}, \cdots, \boldsymbol{A}^{(P)}]\!]_{j_1,\cdots,j_P}, \beta^{-1}).$$

- The prior was introduced in previous slides.

Inference Algorithms for Bayesian TDs

- Under the evidence maximization framework, the inference problem can be formulated as

$$\mathcal{L}(q(\boldsymbol{\theta})) \triangleq \int q(\boldsymbol{\theta}) \log \frac{p(\mathcal{D}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})} d\boldsymbol{\theta}.$$

- Unknown parameters:

$$\boldsymbol{\theta} \triangleq \{\{\boldsymbol{A}^{(p)}\}_{p=1}^{P}, \{\zeta_l\}_{l=1}^{L}, \beta\}.$$

- The joint pdf:

$$p(\mathcal{D}, \boldsymbol{\theta}) \triangleq p(\boldsymbol{\mathcal{Y}}_{\boldsymbol{\Omega}}, \{\{\boldsymbol{A}^{(p)}\}_{p=1}^{P}, \{\zeta_l\}_{l=1}^{L}, \beta\}).$$

- To get over the intractable integration difficulty, the ELBO maximization problem is solved by further constraining $q(\boldsymbol{\theta})$ into a functional family $\mathcal{F}$, i.e., $q(\boldsymbol{\theta}) \in \mathcal{F}$.
- Among all the functional families, the mean-field family is the most widely used one:

$$q(\boldsymbol{\theta}) = \prod_{p=1}^{P} q(\boldsymbol{A}^{(p)}) q(\{\zeta_l\}_{l=1}^{L}) q(\beta).$$

- Inspect the mean-field family:

$$q(\boldsymbol{\theta}) = \prod_{p=1}^{P} q(\boldsymbol{A}^{(p)}) q(\{\zeta_l\}_{l=1}^{L}) q(\beta).$$

- The factorized structure above inspires the idea of block minimization.

- Inspect the mean-field family:

$$q(\boldsymbol{\theta}) = \prod_{p=1}^{P} q(\boldsymbol{A}^{(p)}) q(\{\zeta_l\}_{l=1}^{L}) q(\beta).$$

- The factorized structure above inspires the idea of block minimization.
- After fixing the variational pdfs $\{q(\boldsymbol{\theta}_j)\}_{j \neq k}$, the optimal $q(\boldsymbol{\theta}_k)$ was shown to be:

$$q^*\left(\boldsymbol{\theta}_k\right) = \frac{\exp\left(\mathbb{E}_{\prod_{j \neq k} q(\boldsymbol{\theta}_j)}\left[\ln p\left(\mathcal{D}, \boldsymbol{\theta}\right)\right]\right)}{\int \exp\left(\mathbb{E}_{\prod_{j \neq k} q(\boldsymbol{\theta}_j)}\left[\ln p\left(\mathcal{D}, \boldsymbol{\theta}\right)\right]\right) d\boldsymbol{\theta}_k}.$$

- MF-VI imposes a factorization structure on $q(\boldsymbol{\theta})$, which implies statistical independence of the variables $\boldsymbol{\theta}_k$ given observed dataset $\mathcal{D}$.
- If this is not the case, the mean-field approximation will lead to mismatch when approaching the ground-truth posteriors.

- MF-VI imposes a factorization structure on $q(\boldsymbol{\theta})$, which implies statistical independence of the variables $\boldsymbol{\theta}_k$ given observed dataset $\mathcal{D}$.
- If this is not the case, the mean-field approximation will lead to mismatch when approaching the ground-truth posteriors.
- To achieve more accurate posterior estimation, it is trendy to employ more advanced variational approximation techniques, e.g., Stein VI.
- Practical issues:
  1. informative initial values
  2. computational complexity
  3. low SNR region difficulties
  4. etc.

Outline

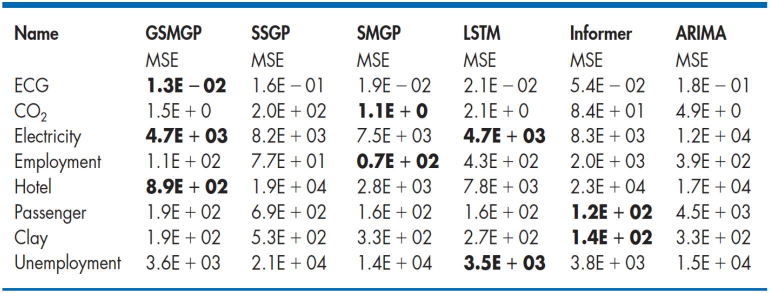
Section Goals

Finally, we showcase the following practical applications:

- Time series prediction using Gaussian process (GP) models;
- Adversarial learning using Bayesian deep neural networks (DNNs);
- Social group clustering and image completion using unsupervised tensor decompositions (TDs).

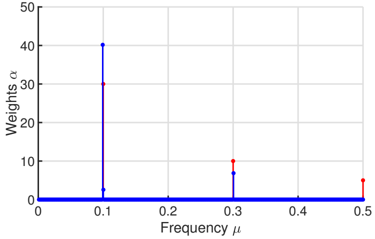
Time Series Prediction via GPs: Classic Datasets

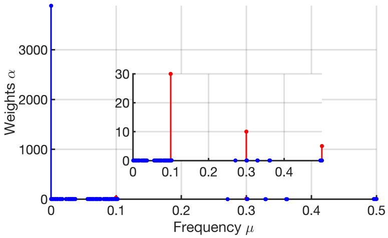
- Consider classic 1D time series datasets with different lengths and evaluate multi-step ahead prediction in terms of MSE.
- Compare the sparsity-promoting GP models (including GSMGP, SSGP, SMGP) with some SOTA time series prediction models.



Name	GSMGP	SSGP	SMGP	LSTM	Informer	ARIMA
	MSE	MSE	MSE	MSE	MSE	MSE
ECG	**1.3E − 02**	1.6E − 01	1.9E − 02	2.1E − 02	5.4E − 02	1.8E − 01
CO_2	1.5E + 0	2.0E + 02	**1.1E + 0**	2.1E + 0	8.4E + 01	4.9E + 0
Electricity	**4.7E + 03**	8.2E + 03	7.5E + 03	**4.7E + 03**	8.3E + 03	1.2E + 04
Employment	1.1E + 02	7.7E + 01	**0.7E + 02**	4.3E + 02	2.0E + 03	3.9E + 02
Hotel	**8.9E + 02**	1.9E + 04	2.8E + 03	7.8E + 03	2.3E + 04	1.7E + 04
Passenger	1.9E + 02	6.9E + 02	1.6E + 02	1.6E + 02	**1.2E + 02**	4.5E + 03
Clay	1.9E + 02	5.3E + 02	3.3E + 02	2.7E + 02	**1.4E + 02**	3.3E + 02
Unemployment	3.6E + 03	2.1E + 04	1.4E + 04	**3.5E + 03**	3.8E + 03	1.5E + 04

- Synthetic data with three modes.
- Sparse solution identifies the most effective frequency components of the data and, thus, leads to good model interpretability.





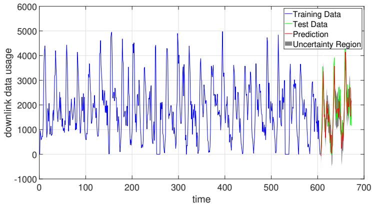
Time Series Prediction via GPs: Real 5G Dataset[18]

- Natural uncertainty region of a point prediction for GP models over deterministic models.

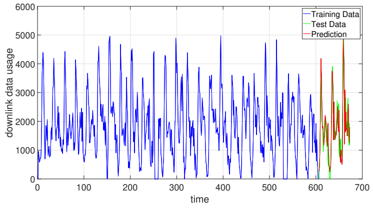
[18] Feng Yin et.al., "Wireless traffic prediction with scalable Gaussian process: Framework, algorithms, and verification", IEEE JSAC, pp.1291-1306,2019.

Time Series Prediction via GPs: Real 5G Dataset[18]

- Natural uncertainty region of a point prediction for GP models over deterministic models.



GSM based GP



Classic LSTM

- GSMGP with $Q = 500$ fixed grids and $e_{\mathrm{MAPE}} = 0.28$; LSTM model with $e_{\mathrm{MAPE}} = 1.12$.
- The gray shaded areas represent the uncertainty region of GP model.

[18] Feng Yin et.al., "Wireless traffic prediction with scalable Gaussian process: Framework, algorithms, and verification", IEEE JSAC, pp.1291-1306,2019.

Adversarial Learning via Bayesian DNNs

- Recent studies have revealed that DNNs are highly susceptible to *adversarial examples*, i.e., cleverly crafted examples whose purpose is that of *fooling* a considered model into *misclassification*.

Adversarial Learning via Bayesian DNNs

- Recent studies have revealed that DNNs are highly susceptible to *adversarial examples*, i.e., cleverly crafted examples whose purpose is that of *fooling* a considered model into *misclassification*.
- For example, projected gradient descent (PGD)-based adversarial example construction.
- Under PGD, the obtained perturbed modification to the original example, $\boldsymbol{x}$, which seems small or even imperceptible to human being, can fool the model to misclassification.

- Drawing upon this vulnerability, adversarial training has been recently devoted towards more reliable and robust DNNs.
- Stochastic modeling rationale: Introducing *stochasticity* into the considered architecture, e.g., by randomizing the input data and/or the learning model itself.
- Bayesian learning offers a natural stochastic defense framework towards more adversarially robust networks.

The so-called doubly stochastic nature stems from:

- Sparsity-promoting link-wise non-parametric IBP prior;
- Stochastic adaptation of the biologically inspired and competition-based LWTA activation.

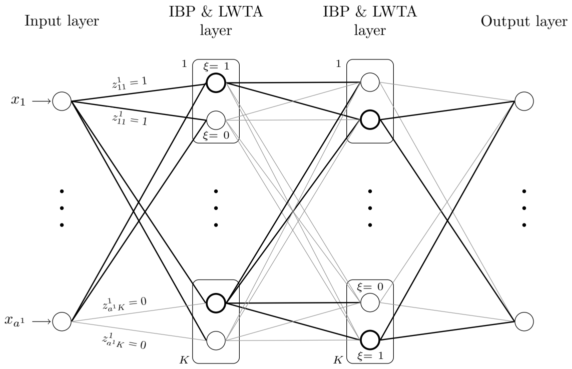


Table: Natural and Robust accuracy under a conventional PGD attack with 20 steps and 0.007 step-size using WideResNet-34 models with different widen factors.

Adversarial Training-PGD

	Natural Accuracy (%)		Robust Accuracy (%)	
Widen Factor	Baseline	Stochastic LWTA	Baseline	Stochastic LWTA
1	74.04	**87.0**	49.24	**81.87**
5	83.95	**91.88**	54.36	**83.4**
10	85.41	**92.26**	55.78	**84.3**

- Baseline: B. Wu *et.al.* "Do wider neural networks really help adversarial robustness?" in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2021.
- Using the same PGD-based Adversarial Training scheme for all models.

Table: Robust Accuracy (%) comparison under the AutoAttack framework.

Method	AutoAttack
HE	53.74
WAR	54.73
Pre-training †	54.92
Data augmentation †	65.88
WAR†	61.84
Stochastic-LWTA/PGD/WideResNet-34-1	**74.71**
Stochastic-LWTA/PGD/WideResNet-34-5	**81.22**
Stochastic-LWTA/PGD/WideResNet-34-10	**82.60**

Table: Robust Accuracy (%) comparison under the AutoAttack framework.

Method	AutoAttack
HE	53.74
WAR	54.73
Pre-training †	54.92
Data augmentation †	65.88
WAR†	61.84
Stochastic-LWTA/PGD/WideResNet-34-1	**74.71**
Stochastic-LWTA/PGD/WideResNet-34-5	**81.22**
Stochastic-LWTA/PGD/WideResNet-34-10	**82.60**

- † denotes models that are trained with additional unlabeled data.
- The AutoAttack performance corresponds to the final robust accuracy after employing all the attacks in AA.

Unsupervised Learning via Bayesian TDs

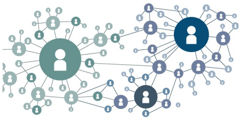
- We present two unsupervised learning applications:
  1. Social group clustering: adopts Bayesian tensor CPD;
  2. Image completion: employs Bayesian tensor TTD.
- Those algorithms bypass the need of hyper-parameters tuning and effectively avoid overfitting.

Bayesian Tensor CPD for Social Group Clustering

- Dataset: *ENRON E-mail corpus* dataset (a 3-D tensor with the size $184 \times 184 \times 44$ ($\sharp$ of sender, $\sharp$ of receiver, $\sharp$ of day))

Bayesian Tensor CPD for Social Group Clustering

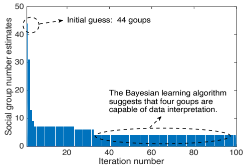
- Dataset: *ENRON E-mail corpus* dataset (a 3-D tensor with the size $184 \times 184 \times 44$ ($\sharp$ of sender, $\sharp$ of receiver, $\sharp$ of day))
- We aim to demonstrate how the Bayesian tensor CPD can be used to simultaneously
    - determine the number of social groups (automatic tensor rank determination),
    - cluster people into different groups (interpretable source separation),
    - extract interpretable temporal profiles of different social groups (interpretable source separation).



Social group clustering

ENRON e-mail corpus

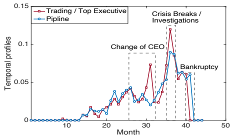
Bayesian tensor CPD



Automatically determine the social group number

Legal	Government Affair
Trading / Top Executive	Pipeline

Clustering results (name list was given in Table V of [9])



Interpretable temporal profiles

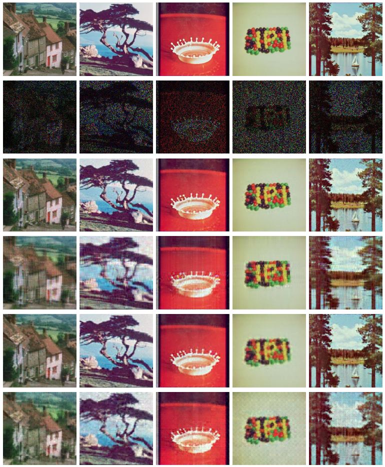
Bayesian Tensor TTD for Image Completion

- Color images are naturally 3-D tensor.
- Fold an image into a higher dimensional tensor (e.g., 9-D tensor), and then apply TTD to recover the missing pixels.

Bayesian Tensor TTD for Image Completion

- Color images are naturally 3-D tensor.
- Fold an image into a higher dimensional tensor (e.g., 9-D tensor), and then apply TTD to recover the missing pixels.
- For a P-D tensor, TTD has $P - 1$ hyper-parameters (called TT ranks).
- Manually tuning different combinations of these hyper-parameters for overfitting avoidance is time-consuming.
- Bayesian TTD, using the ideas introduced before, can automatically learn the most suitable TT-ranks to match the underlying image pattern.

- Experimental results for visual comparison on image completion with 80% missing data.
- Ground-truth images are in the top row.
- The second row includes the images with missing values.
- The third to the bottom rows include results from Bayesian TTD, TTC-TV, TMAC-TT, and STTO.



Outline

Conclusion

- Presented a gentle overview of state-of-the-art sparsity-promoting priors for both Bayesian parametric and nonparametric models.
- Constrained our focus to modern GP models, Bayesian DNNs, and TDs.
- Introduced advanced prior design strategies and inference algorithms.
- Demonstrated a wide spectrum of signal processing and machine learning applications.

The reported results indicated that:

1. Sparsity-promoting priors are adaptive to varying data and enable automatic model structure selection;
2. Sparsity-promoting priors lead to natural and reasonable uncertainty quantification;
3. Sparse solution can better reveal the underlying characteristics of a target system/signal with the most effective components.

Future Directions

Potential challenges include but not limited to:

- Quality of the posterior/predictive distribution, compared with that provided by conformal prediction.
- Issues with misspecification of the learning model and noise statistics.
- Emerging sparsity-promoting mechanisms inspired from neuroscience.
- More emerging applications in complex systems, such as 6G, ISAC, autonomous driving system, ocean sensing, etc.
- Sparsity-awareness in emerging learning paradigms and large language models (LLMs).

Thanks for your attention!