PML 9. Generative Models: A Guide Into the literature

Probabilistic Machine Learning Reading Group

David Rios Insua

March 11 2026

ICMAT-CSIC

Contents

- Generative Models: Basic Concepts
- Variational autoencoders
- Autoregressive models
- Final comments

Generative Models: Basic Concepts

GMs: Basic concepts

- ML: Generative vs Discriminative
- GM. Joint probability distribution for $p(\mathbf{x})$ for $\mathbf{x} \in \mathcal{X}$.
- Conditional GM. Conditional on inputs $\mathbf{c} \in \mathcal{C}$, $p(\mathbf{x}|\mathbf{c})$.
  1 correct y vs several correct y

Types

- Deep generative models. Use NN to learn mapping from latent z to data x
- Probabilistic graphical models. See slides Session 1. Maps interconnected latents $z_1, ..., z_N$ to observed $x_1, ..., x_D$
- All kinds of 'oldies': STATS-101, mixture models, KDEs,...

Goals of GMs. I

- Generating data (Generative AI)
  `https://llms-cunef-icmat-rg2024.github.io/`
  - Sample new images.
  - Synthetic data.
  - Conditional generation
    - $\mathbf{c}=$ text prompt, $\mathbf{x}=$ image, text-2-image
    - image-2-text (captioning), image-2-image (colorization), speech-2-text (ASR), sequence-2-sequence (machine translation,text generation).
- Density estimation $p(x)$
  - Outlier detection, Novelty detection, Distribution shift, mixture modeling,...

Goals of GMs. II

- Multiple imputation $p(x_m|x_o)$
- Structure discovery $p(z|x) \propto p(z)p(x|z)$
- Model-based reinforcement learning, when data is not abundant
- Data compression
- Decision support, Decision making

Unsupervised learning...

Deep GMs

- Variational Autoencoders
- Autoregressive models
- Normalizing flows (sess 10)
- Energy models (sess 11)
- Diffusion models (sess 12)
- Generative adversarial networks

Murphy's chs 20, 21, 22 (and other along the slides)

Features of GMs

- Density available. Explicit, only implicit, lower bound on density, approximation.
- Sampling. Fast(VAE) or slow (ARM).
- Training. MLE (local, global), MAP, posterior, min-max,...
- Presence of latents. Yes, No, Compressed.
- Imposed architecture restrictions (eg invertible neural nets)

Evaluating GMs

Metrics capturing

- Sample quality: Samples generated by model part of data distribution?
- Sample diversity: Samples capture all modes?
- Generalization. Model generalizes beyond training data?

Evaluating GMs

- KL divergence between the model q and the actual p

$$D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx = -H(p) + H_{ce}(p, q)$$

  Approach $p(x)$ by empirical distribution, cross entropy through NLL on dataset $-\frac{1}{N} \sum_n \log q(x_n)$
- Two sample tests. Null: Both sample from same distribution. Classical tests adapted to high dimension.
- Human evaluation, e.g. in images.
- ....

GMs refs

- Murphy (2023) PML, Chap.20
- Tomczak (2022) Deep GMs
- Gallego, DRI (2022) Recent developments in NNs.

Variational autoencoders

VAEs

- Definition
- Autoencoders
- Basics
- Variants

VAE. Definition

GMs with latent variables z, a prior over z, a deep NN decoder $d_\theta(z)$ for the x, and an exponential family transformation

$$z \sim p_\theta(z)$$
$$x|z \sim Expfam(x|d_\theta(z))$$

Expfam designates a distribution of the exponential familiy (Gaussian, product of Bernoullis,....)

Deep latent variable model (DLVM).
When prior Gaussian, Deep latent Gaussian model (DLGM)

Computing $p_\theta(z|x)$ (and $p_\theta(x)$) is computationally intractable...

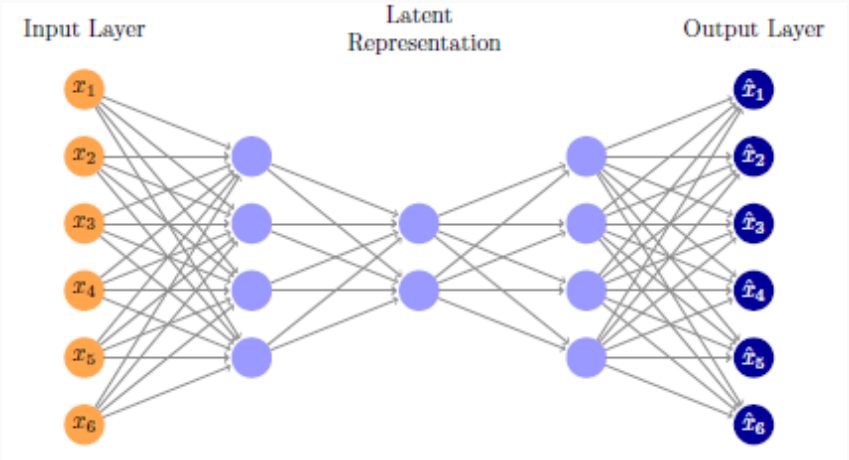
Hence VI through $q_\phi(z|x)$, recognition or inference network, to do approximate posterior inference (Sess 2, 5)

Autoencoder

A NN model that maps inputs x to a low-dimensional latent space using an encoder, $z = f_e(x)$, and then attempts to reconstruct the inputs using a decoder, $x = f_d(z)$.

Trained to minimize $L(\theta) = ||f_d(f_e(x)) - x||^2$.

Once trained, ignore reconstructed x and use z as required.



A 'nonlinear PCA'

VAE. Basics

VAEs are probabilistic autoencoders

- AE objective function log likelihood, no KL term
- Encoding is deterministic. Encoder network computes $E(z|x)$ not $V(z|x)$

VAE. Basics

A basic VAE defines $p_\theta(x, z) = p_\theta(z)p_\theta(x|z)$ with $p_\theta(z)$ Gaussian; $p_\theta(x|z)$ product of ExpFam distributions with parameters defined through a NN decoder $d_\theta(z)$. For example for binary observations, use

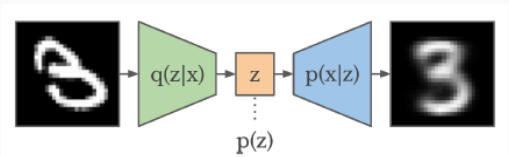
$$p_\theta(x|z) = \prod_{d=1}^{D} Ber(x_d|\sigma(d_\theta(z)))$$

Also fits a recognition model to perform approximate posterior inference

$$q_\phi(z, x) = q(z|e_\phi(x)) \simeq p_\theta(z|x)$$

A typical example $q_\phi(z|x) \sim N(z|\mu, diag(\exp(l))$

$$(\mu, l) = e_\phi(x)$$



VAE. Basics

Recall Stochastic VI (session 2)

Data:

$\mathcal{D}$: Dataset

$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: Inference model

$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

$\boldsymbol{\theta}, \boldsymbol{\phi}$: Learned parameters

$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Initialize parameters

while *SGD not converged* **do**

  $\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

  $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every datapoint in $\mathcal{M}$)

  Compute $\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$ and its gradients $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$

  Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using SGD optimizer

end

Amortized Stochastic VI

Algorithm 21.1: Fitting a VAE with Bernoulli likelihood and full covariance Gaussian posterior. Based on Algorithm 2 of [KW19a].

Initialize $\boldsymbol{\theta}$, $\boldsymbol{\phi}$
repeat
  Sample $\boldsymbol{x} \sim p_{\mathcal{D}}$
  Sample $\boldsymbol{\epsilon} \sim q_0$
  $(\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}') = e_{\boldsymbol{\phi}}(\boldsymbol{x})$
  $\mathbf{M} = \text{np.triu}(\text{np.ones}(K), -1)$
  $\mathbf{L} = \mathbf{M} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma})$
  $\boldsymbol{z} = \mathbf{L}\boldsymbol{\epsilon} + \boldsymbol{\mu}$
  $p_{\boldsymbol{\theta}} = d_{\boldsymbol{\theta}}(\boldsymbol{z})$
  $\mathcal{L}_{\text{logqz}} = -\sum_{k=1}^{K} \left[\frac{1}{2}\epsilon_k^2 + \frac{1}{2}\log(2\pi) + \log \sigma_k\right]$ // from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ in Equation (10.47)
  $\mathcal{L}_{\text{logpz}} = -\sum_{k=1}^{K} \left[\frac{1}{2}z_k^2 + \frac{1}{2}\log(2\pi)\right]$ // from $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ in Equation (10.48)
  $\mathcal{L}_{\text{logpx}} = -\sum_{d=1}^{D} \left[x_d \log p_d + (1 - x_d)\log(1 - p_d)\right]$ // from $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$
  $\mathcal{L} = \mathcal{L}_{\text{logpx}} + \mathcal{L}_{\text{logpz}} - \mathcal{L}_{\text{logqz}}$
  Update $\boldsymbol{\theta} := \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}$
  Update $\boldsymbol{\phi} := \boldsymbol{\phi} - \eta \nabla_{\boldsymbol{\phi}} \mathcal{L}$
until converged

VAE. Variants

- Beta-VAE $\mathcal{L}_\beta(\theta, \phi|x) = \underbrace{-\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right]}_{\mathcal{L}_E} + \beta \underbrace{D_{\mathrm{KL}}\left(q_\phi(z|x) \parallel p_\theta(z)\right)}_{\mathcal{L}_R}$
- Info-VAE $Ł(\theta, \phi|x) = -\lambda D_{\mathrm{KL}}\left(q_\phi(z) \parallel p_\theta(z)\right) - \mathbb{E}_{q_\phi(z)}\left[D_{\mathrm{KL}}\left(q_\phi(x|z) \parallel p_\theta(x|z)\right)\right] + \alpha\, \mathbb{I}_q(x; z)$
- Mutimodal VAE. Several modes of data (e.g. text and images)
- Hierarchical VAE. With several stochastic layers
- .....

VAE REFs

- Murphy (2023) PML, Chap.21
- Kingma, Welling (2013) Auto-encoding variational Bayes
- Kingma, Welling (2019) An introduction to variational autoencoders

Autoregressive models

Autoregressive models

- Basic definitions
- NNs for autoregressions
- Transformers

Basic models

Autoregressive model (ARM), using chain rule

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)...p(x_T|x_1, ..., x_{T-1})$$

Conditioning on c

$$p(x_{1:T}|c) = p(x_1|c)p(x_2|x_1, c)p(x_3|x_2, x_1, c)...p(x_T|x_1, ..., x_{T-1}, c)$$

Taming the beast...

- (First order) Markov assumption
  $p(x_t|x_1, ..., x_{t-1}) = p(x_t|x_{t-1})$ AR(1)
- (Higher order) Markov assumptions.
- x_t depends on z_t, which compresses the past.
  - If z_t is deterministic function of x_{t-1} RNN
  - If z_t is stochastic function of z_{t-1} HMM

NNs for autoregression

Use a functional form for the conditionals

like a (shallow) NN

$$p(y_1, y_2, \ldots, y_N) = p(y_1, \ldots, y_q) \prod_{t=q+1}^{N} p(y_t \mid y_{t-1}, y_{t-2}, \ldots, y_{t-q})$$

$$x_t = (y_{t-1}, \ldots, y_{t-q});$$

$$\widehat{y}_t(x_t) = \sum_{j}^{M} \beta_j \varphi(x_t' \gamma + \delta_j)$$

Nonlinear AR(q) model

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-q}) + \epsilon_t, \; t = q+1, \ldots, N$$
$$\epsilon_t \sim N(0, \sigma^2),$$

Linear AR(q)+Nonlinear AR(q)

$$f(y_{t-1}, y_{t-2}, \ldots, y_{t-q}) \equiv x_t' \lambda + \sum_{j}^{M} \beta_j \varphi(x_t' \gamma), \; t = q+1, \ldots, N$$

NNs for autoregression

Priors

$$\beta_j \sim N(\mu_\beta, \sigma_\beta^2), \quad \lambda \sim N(\mu_\lambda, \sigma_\lambda^2 I)$$

$$\gamma_j \sim N(\mu_\gamma, \Sigma_\gamma), \quad \sigma^2 \sim InvGamma(a_\sigma, b_\sigma).$$

$$\mu_\beta \sim N(a_{\mu_\beta}, b_{\mu_\beta}), \quad \sigma_\beta^2 \sim InvGamma(a_{\sigma_\beta}, b_{\sigma_\beta})$$

$$\mu_\lambda \sim N(a_{\mu_\lambda}, b_{\mu_\lambda}), \quad \sigma_\lambda^2 \sim InvGamma(a_{\sigma_\lambda}, b_{\sigma_\lambda})$$

$$\mu_\gamma \sim N(a_{\mu_\gamma}, b_{\mu_\gamma}), \quad \Sigma_\gamma \sim InvWishart(a_{\Sigma_\lambda}, b_{\Sigma_\lambda})$$

NNs for autoregression. Sampler

1. Given current values of χ and σ^2 (β and λ are marginalized), for each γ_j, $j = 1, \ldots, M$, generate a proposal $\widetilde{\gamma}_j \sim N(\gamma_j, c\Sigma_\gamma)$, calculate the acceptance probability,

$$a = \min\left[1, \frac{p(\widetilde{\gamma} \mid \mu_\gamma, \Sigma_\gamma)p(D' \mid y_1, \ldots, y_q, \widetilde{\gamma}, \sigma^2)}{p(\gamma \mid \mu_\gamma, \Sigma_\gamma)p(D' \mid y_1, \ldots, y_q, \gamma, \sigma^2)}\right]$$

where $\gamma = (\gamma_1, \ldots, \gamma_{j-1}, \gamma_j, \gamma_{j+1}, \ldots, \gamma_M)$, and $\widetilde{\gamma} = (\gamma_1, \ldots, \gamma_{j-1}, \widetilde{\gamma}_j, \gamma_{j+1}, \ldots, \gamma_M)$. With probability a, replace γ by $\widetilde{\gamma}$ and rearrange indices if necessary to satisfy order constraint. Otherwise, leave γ_j unchanged.

2. Generate new values for parameters, drawing from their full conditional posteriors:

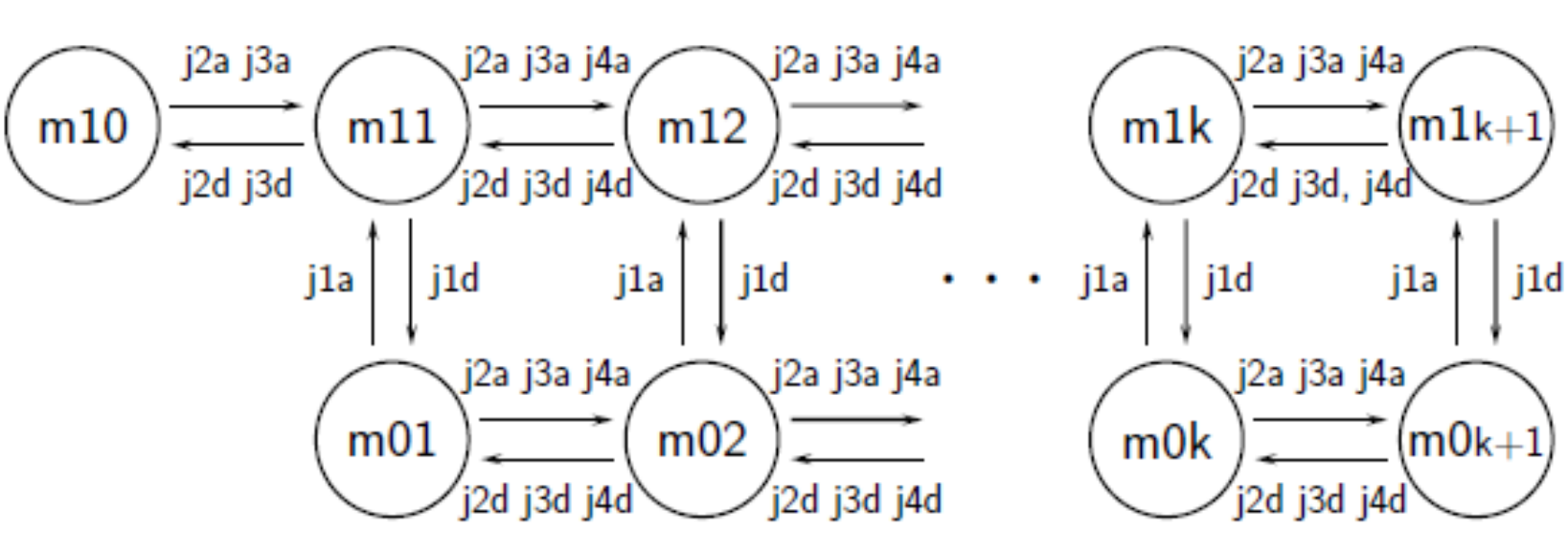
$$\widetilde{\beta} \sim p\left(\beta \mid D', \gamma, \lambda, \sigma^2, \chi\right) \text{ is a multivariate normal distribution.}$$
$$\widetilde{\lambda} \sim p\left(\lambda \mid D', \gamma, \beta, \sigma^2, \chi\right) \text{ is a multivariate normal distribution.}$$
$$\widetilde{\sigma^2} \sim p\left(\sigma^2 \mid D', \gamma, \beta, \lambda\right) \text{ is an inverse gamma distribution.}$$

3. Given current values of $(\gamma, \beta, \lambda, \sigma^2)$, generate a new value for each hyperparameter by drawing from their complete conditional posterior distributions:

$$\widetilde{\mu_\beta} \sim p\left(\mu_\beta \mid D', \beta, \sigma_\beta^2\right) \text{ is a normal distribution.}$$
$$\widetilde{\sigma_\beta^2} \sim p\left(\sigma_\beta^2 \mid D', \beta, \mu_\beta\right) \text{ is an inverse gamma distribution.}$$
$$\widetilde{\mu_\lambda} \sim p\left(\mu_\lambda \mid D', \lambda, \sigma_\lambda^2\right) \text{ is a multivariate normal distribution.}$$

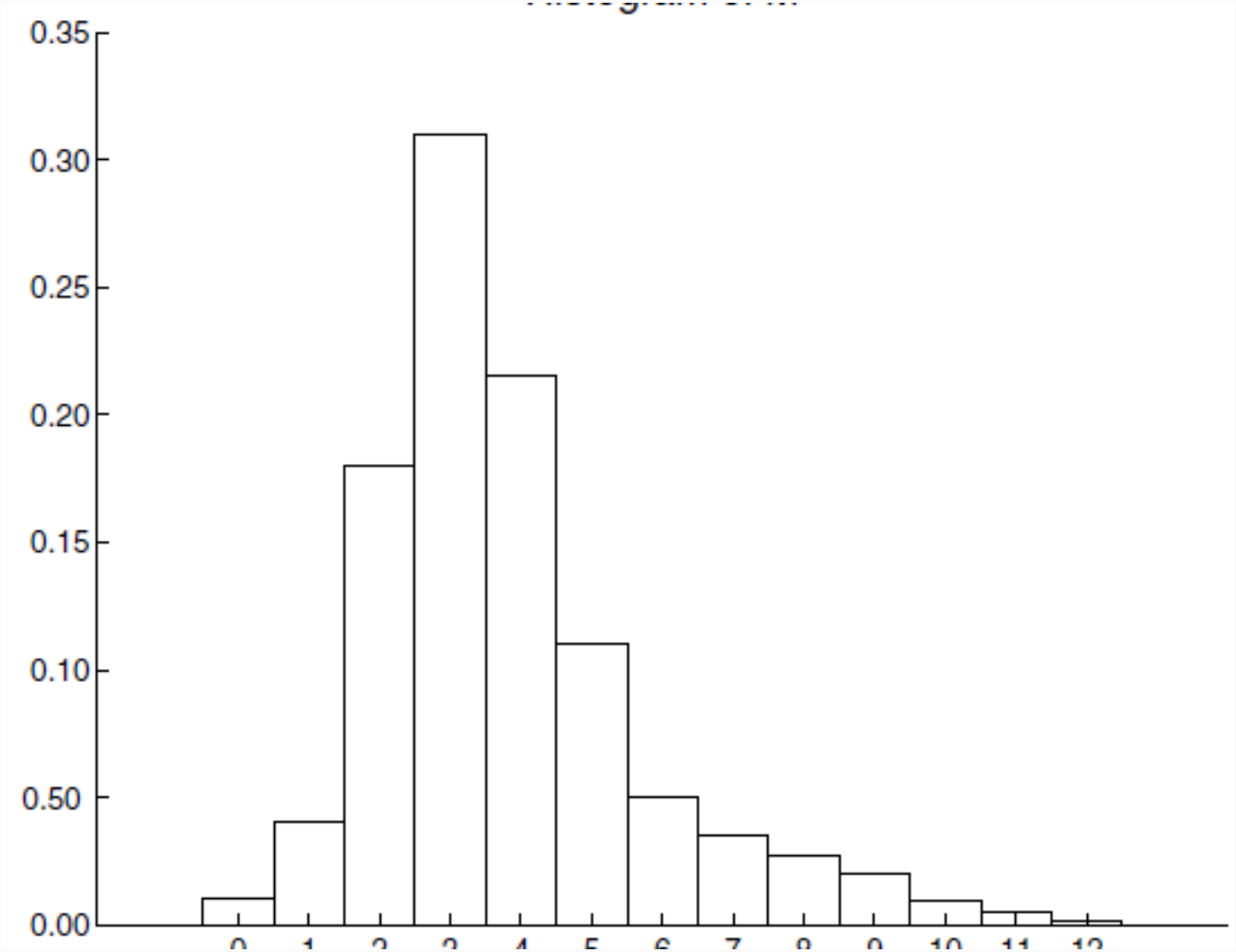
$$\widetilde{\sigma_\lambda^2} \sim p\left(\sigma_\lambda^2 \mid D', \lambda, \mu_\lambda\right) \text{ is an inverse gamma distribution.}$$
$$\widetilde{\mu_\gamma} \sim p\left(\mu_\gamma \mid D', \gamma, \Sigma_\gamma\right) \text{ is a multivariate normal distribution.}$$
$$\widetilde{\Sigma_\gamma} \sim p\left(\Sigma_\gamma \mid D', \gamma, \mu_\gamma\right) \text{ is an inverse Wishart distribution.}$$

NNs for autoregression. RJ



- Add or delete linear term
- Add or delete linearized node
- Add or delete irrelevant nodes
- Add or delete duplicate nodes

NNs for autoregression. RJ Issues



NNs for autoregression. Sampler. Inefficiencies for large problems...

1. Given current values of χ and σ^2 (β and λ are marginalized), for each γ_j, $j = 1, \ldots, M$, generate a proposal $\widetilde{\gamma}_j \sim N(\gamma_j, c\Sigma_\gamma)$, calculate the acceptance probability,

$$a = \min\left[1, \frac{p(\widetilde{\gamma} \mid \mu_\gamma, \Sigma_\gamma)p(D' \mid y_1, \ldots, y_q, \widetilde{\gamma}, \sigma^2)}{p(\gamma \mid \mu_\gamma, \Sigma_\gamma)p(D' \mid y_1, \ldots, y_q, \gamma, \sigma^2)}\right]$$

where $\gamma = (\gamma_1, \ldots, \gamma_{j-1}, \gamma_j, \gamma_{j+1}, \ldots, \gamma_M)$, and $\widetilde{\gamma} = (\gamma_1, \ldots, \gamma_{j-1}, \widetilde{\gamma}_j, \gamma_{j+1}, \ldots, \gamma_M)$. With probability a, replace γ by $\widetilde{\gamma}$ and rearrange indices if necessary to satisfy order constraint. Otherwise, leave γ_j unchanged.

2. Generate new values for parameters, drawing from their full conditional posteriors:

$$\widetilde{\beta} \sim p\left(\beta \mid D', \gamma, \lambda, \sigma^2, \chi\right) \text{ is a multivariate normal distribution.}$$
$$\widetilde{\lambda} \sim p\left(\lambda \mid D', \gamma, \beta, \sigma^2, \chi\right) \text{ is a multivariate normal distribution.}$$
$$\widetilde{\sigma^2} \sim p\left(\sigma^2 \mid D', \gamma, \beta, \lambda\right) \text{ is an inverse gamma distribution.}$$

3. Given current values of $(\gamma, \beta, \lambda, \sigma^2)$, generate a new value for each hyperparameter by drawing from their complete conditional posterior distributions:

$$\widetilde{\mu_\beta} \sim p\left(\mu_\beta \mid D', \beta, \sigma_\beta^2\right) \text{ is a normal distribution.}$$
$$\widetilde{\sigma_\beta^2} \sim p\left(\sigma_\beta^2 \mid D', \beta, \mu_\beta\right) \text{ is an inverse gamma distribution.}$$
$$\widetilde{\mu_\lambda} \sim p\left(\mu_\lambda \mid D', \lambda, \sigma_\lambda^2\right) \text{ is a multivariate normal distribution.}$$

$$\widetilde{\sigma_\lambda^2} \sim p\left(\sigma_\lambda^2 \mid D', \lambda, \mu_\lambda\right) \text{ is an inverse gamma distribution.}$$
$$\widetilde{\mu_\gamma} \sim p\left(\mu_\gamma \mid D', \gamma, \Sigma_\gamma\right) \text{ is a multivariate normal distribution.}$$

NNs for autoregression

- Neural autoregressive density estimators
- Causal CNNs
- Transformers

Transformers

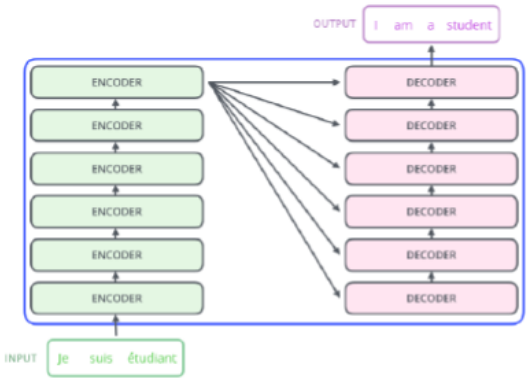
- Encoding, Decoding, Encoding-Decoding.

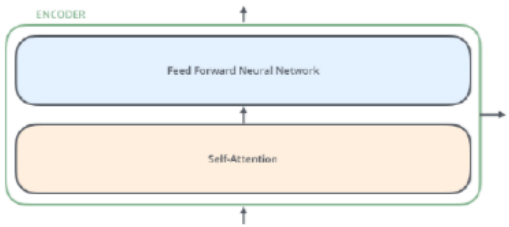
Decoder (Generator) works as follows:
For each tranformer layer

- For each $y_{1:t}$ apply attention to compute attention weights $a_{1:t}$.
- Compute activation $z_t = \sum a_\tau y_\tau$
- Pass through feed-forward layer $h_t = MLP(z_t)$

Output $y_{t+1} = Cat(softmax(Wh_t))$

Transformers





Embedding of i-th token x_i

Query vector $q_i = x_i' \gamma_q$

Key vector $k_i = x_i' \gamma_k$

Value vector $v_i = x_i' \gamma_v$

Output $\text{softmax}\left(\frac{qk'}{\sqrt{d_k}}\right) v$

Transformers

Full details with code in Session 2

`https://llms-cunef-icmat-rg2024.github.io/`

How this is used in GPTs in Session 3

`https://llms-cunef-icmat-rg2024.github.io/`

ARM REFs

- Murphy (2023) PML, Chap.22
- Menchero, et al (2005) Bayesian Analysis of Nonlinear Autoregression Models Based on NNs, Neur. Comp.
- Prado and West (2010) Time series: modeling, computation and inference
- Vaswani Attention is all you need. Advances in NIPS 30 (2017).
- Zhao, Wayne Xin, et al. A survey of large language models. arXiv:2303.18223 1.2 (2023): 1-124.
- `https://llms-cunef-icmat-rg2024.github.io/`
- Papamarkou et al. Position: Bayesian deep learning is needed in the age of large-scale AI. arXiv:2402.00809 (2024).

Final comments

Final comments

- Several of these models actually lack a PML approach
- Attacks on LLMs lagging
- Attacks on unsupervised learning lagging

Final comments. Next sessions

- Normalizing flows
- Energy Models
- Diffusion models

Questions?