Time Series Models: From Statistics to AI

lecturer: Daniel Durstewitz
tutors: Lukas Eisenmann, Christoph Hemmer, Alena Brändle, Florian Hess, Elias Weber, **Florian Götz**
SS2025

Exercise Sheet 9

Regulations

Please submit your solutions via Moodle in teams of **2 students**, before the exercise group on **Wednesday, April 30th, 2025**. Each submission must include **exactly** two files:

- A `.pdf` file containing both your Jupyter notebook and solutions to analytical exercises. The Jupyter notebook can be exported to pdf by selecting **File → Download as → pdf** in JupyterLab. If this method does not work, you may print the notebook as a pdf instead. Your analytical solutions can be either scanned handwritten solutions or created using LaTeX.
- A `.ipynb` file containing your code as Jupyter notebook.

Both files must follow the naming convention:
`Lastname1-Lastname2-sheet09.pdf`
`Lastname1-Lastname2-sheet09.ipynb`

1 The Reparameterization Trick in Variational Inference

Consider a latent variable model with observed data $x \in \mathbb{R}^d$ and latent variable $z \in \mathbb{R}^k$. We approximate the true posterior $p(z \mid x)$ using a variational distribution $q_\phi(z \mid x)$, chosen to be a diagonal Gaussian:

$$q_\phi(z \mid x) = \mathcal{N}(z; \mu_\phi(x), \mathrm{diag}(\sigma_\phi^2(x)))$$

where $\mu_\phi(x), \sigma_\phi(x) \in \mathbb{R}^k$ are outputs of a neural network with parameters ϕ. The generative model is defined via a likelihood $p_\theta(x \mid z)$, with parameters θ, and a standard normal prior $p(z) = \mathcal{N}(0, I)$.

We optimize the evidence lower bound (ELBO):

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - D_{\mathrm{KL}}(q_\phi(z \mid x) \parallel p(z))$$

(a) **Gradient issue**
Explain why computing the gradient

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)]$$

is problematic due to the dependency of the sampling distribution on ϕ. Why does this pose a challenge for gradient-based optimization?

(b) **Derivation of the reparameterization trick**
Let $\epsilon \sim \mathcal{N}(0, I)$, and define the transformation

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$$

where $\odot$ denotes elementwise multiplication. Show that:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)}[\log p_\theta(x \mid z(\epsilon, \phi))]$$

and explain why this form allows gradients with respect to ϕ to be computed via the chain rule.

(c) **Extension to sequential latent variables**
Consider a time series model with latent variables $z_1, \ldots, z_T$ and observations $x_1, \ldots, x_T$, with a variational distribution of the form:

$$q_\phi(z_{1:T} \mid x_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t \mid x_{\leq t})$$

and a latent prior given by a first-order Markov model:

$$p(z_1) = \mathcal{N}(0, I), \quad p(z_t \mid z_{t-1}) = \mathcal{N}(Az_{t-1}, \Sigma)$$

Discuss how the reparameterization trick can be applied in this sequential setting. Can each z_t be reparameterized independently? How does the temporal structure of the prior affect this?

2 Reservoir Computing

In this exercise we want to explore the Reservoir Computing (RC) paradigm for DSR. To this end, we will implement an Echo-State Network (ESN). ESNs can be seen as RNNs with fixed dynamical weights and a trainable linear observation model. The ESN we will look at takes the form

$$\begin{aligned} r_t &= (1 - \alpha) r_{t-1} + \alpha \tanh(W r_{t-1} + W_{in} x_t + b) \\ \hat{y}_t &= W_{out} r_t \end{aligned} \tag{1}$$

where $\theta = \{\alpha, W, W_{in}, b\}$ are drawn randomly and stay fixed during training. To train the ESN, we will first drive the reservoir, i.e. we will supply the training time series $X = x_{1:T} \in \mathbb{R}^{T \times N}$ to the ESN to generate a series of reservoir states $R = r_{1:T} \in \mathbb{R}^{T \times M}$. We then use Ridge Regression with regularization parameter λ and cost/loss function

$$L_{RR} = \|Y - R W_{out}^T\|_F^2 + \lambda \|W_{out}\|_F^2, \tag{2}$$

where $Y = x_{2:T+1}$ are the regression targets, to compute W_{out} with closed-form solution

$$W_{out} = Y^T R \left(R^T R + \lambda I\right)^{-1}. \tag{3}$$

To generate from the ESN after training, we simply use eqs. (I) and after an initial warm-up phase of driving the system with data, we feed the reservoir its own predictions (i.e. we replace x_t by $\hat{y}_{t-1}$).

You'll find a basic outline of an ESN implementation in the file `sheet09_template.ipynb`. The file `lorenz_data.npy` contains a Lorenz dataset.

(a) In the template code, implement all necessary functions to train and generate from an ESN. To this end, implement the functions and snippets that say "`your code here`".

(b) Train and fit the ESN with the specified hyperparameters. Generate a trajectory from the model and plot 3D state space. If your ESN implementation is correct, you should get a decent fit of the dynamics.

(c) How low can you go with the latent dimension? How robust is the training, i.e., how much do results differ between networks trained with different sampled reservoir network parameters? (Train five models per M.)