Topic 2A: Forecasting in Time Series: A Mathematical Formulation

Victor M. Preciado

Contents

1 The Forecasting Problem

Time series forecasting is a fundamental problem in statistics and data science, with the primary objective of predicting future values of a sequence based on its historical data. Suppose the current time index is k. Given a stochastic process $\mathcal{Y}$, the forecasting problem specifically involves estimating the next value, Y_{k+1}, based on all available observations up to time k, which we represent with the information set $\mathcal{F}_k$.

The essence of forecasting lies in understanding the probabilistic structure that governs the next observation given the historical data. This structure is encapsulated in the conditional distribution of Y_{k+1} given $\mathcal{F}_k$, i.e.,

$$F_{Y_{k+1}|\mathcal{F}_k}(y \mid \mathcal{F}_k) = \mathbb{P}(Y_{k+1} \leq y \mid \mathcal{F}_k).$$

This conditional distribution, known as the **one-step-ahead forecast distribution**, fully captures the uncertainty surrounding the next value of the time series. It provides a comprehensive description of all possible outcomes for Y_{k+1} given the past data, along with their associated probabilities, thus forming the foundation for probabilistic forecasting in time series analysis

In practical forecasting, the task involves selecting a point estimate $\widehat{y}_{k+1}$ from the forecast distribution that best predicts Y_{k+1} given the information set $\mathcal{F}_k$. The choice of this estimate hinges on the metric chosen to measure the quality of the estimate, as different metrics emphasize various aspects of forecast accuracy. The selection of an appropriate error metric is crucial, as it defines the criteria for evaluating the quality of predictions and directly shapes the forecasting strategy.

1.1 Loss Functions

Given a point estimate $\widehat{y}_{k+1}$ and the actual value y_{k+1} taken by the random variable Y_{k+1}, we quantify the discrepancy between the forecasted and observed values using a **loss function**, denoted by $\ell(y_{k+1}, \widehat{y}_{k+1})$. The choice of loss function influences which aspects of forecast errors are prioritized. The most widely used loss functions in time series forecasting are:

- **Squared Error (SE)**: The Squared Error is one of the most commonly used loss functions due to its mathematical simplicity and differentiability, making it suitable for optimization techniques. The SE quantifies the discrepancy between the forecasted and actual values by squaring their difference, as follows:

  $$\mathrm{SE}(y_{k+1}, \widehat{y}_{k+1}) = (y_{k+1} - \widehat{y}_{k+1})^2.$$

  The squaring operation penalizes larger errors severely, making SE particularly sensitive to outliers. This property is useful when one wants to minimize

significant deviations between forecasts and actual outcomes, including outliers.

- **Absolute Error (AE)**: The Absolute Error measures the magnitude of the discrepancy between forecasted and actual values, without considering the sign of the error, as follows:

$$\mathrm{AE}(y_{k+1}, \widehat{y}_{k+1}) = |y_{k+1} - \widehat{y}_{k+1}|.$$

  AE is less sensitive to outliers compared to SE, as it avoids inflating large errors through squaring. It captures the typical magnitude of forecast errors, making it relevant in scenarios where moderate errors are acceptable, but extreme values are undesirable.

- **Scaled Errors**: To address some limitations of the Absolute Error (AE) and Squared Error (SE), particularly when comparing forecast accuracy across different components or scales, we introduce scaled versions of these loss functions. The core idea is to scale the loss functions, resulting in a **dimensionless metric** that evaluates forecast accuracy relative to the inherent variability of the time series. We define the **Scaled Absolute Error** (SAE) and the **Scaled Squared Error** (SSE) as follows:

$$\mathrm{SAE}(y_{k+1}, \widehat{y}_{k+1}; \mathcal{F}_k) = \frac{|y_{k+1} - \widehat{y}_{k+1}|}{\frac{1}{k-1}\sum_{i=2}^{k} |y_i - y_{i-1}|},$$

$$\mathrm{SSE}(y_{k+1}, \widehat{y}_{k+1}; \mathcal{F}_k) = \frac{(y_{k+1} - \widehat{y}_{k+1})^2}{\frac{1}{k-1}\sum_{i=2}^{k} (y_i - y_{i-1})^2}.$$

  These scaled loss functions normalize the errors by the average absolute or squared differences between consecutive observations, making them robust against variations in scale and ensuring that accuracy assessments are meaningful across different components of the time series.

The choice of a particular error function reflects the priorities of the forecasting task. For instance, SE is particularly suitable when large deviations are critically penalized, while AE provides a more balanced assessment of average error magnitude, especially when the presence of outliers are less of a concern. These functions provide not only a quantitative measure of forecast quality but also shape the development and refinement of forecasting models, ensuring that predictions are as reliable and precise as possible given the available information.

1.2 Conditional Risk

Given a loss function $\ell(y, \widehat{y}_{k+1})$ and a forecast distribution, the predictive **conditional risk** (also known as one-step-ahead **conditional expected loss**) quantifies

the expected error of the forecast estimate $\widehat{y}_{k+1}$ over all possible outcomes of the random variable Y_{k+1}, conditioned on all available information up to time k. Formally, the conditional risk is defined as:

$$\mathcal{R}(\widehat{y}_{k+1}; \mathcal{F}_k) = \mathbb{E}[\ell(Y_{k+1}, \widehat{y}_{k+1}) \mid \mathcal{F}_k] = \int_{-\infty}^{\infty} \ell(y, \widehat{y}_{k+1})\, f_{Y_{k+1}|\mathcal{F}_k}(y \mid \mathcal{F}_k)\, dy, \tag{1}$$

where $\mathbb{E}[\cdot \mid \mathcal{F}_k]$ represents the conditional expectation given the information set $\mathcal{F}_k$, and $f_{Y_{k+1}|\mathcal{F}_k}$ is the probability density function of the one-step-ahead forecast distribution. The conditional risk captures the expected discrepancy between the forecast $\widehat{y}_{k+1}$ and the actual outcome Y_{k+1}, accounting for the probabilities of all possible realizations of Y_{k+1}. This expected loss serves as a comprehensive measure of forecast accuracy by integrating forecast errors over the entire distribution of possible outcomes.

1.3 Optimal Forecast

The goal of forecasting is to select the value of $\widehat{y}_{k+1}$ that minimizes the conditional risk. Assuming knowledge of the forecast density $f_{Y_{k+1}|\mathcal{F}_k}(y \mid \mathcal{F}_k)$, the optimal forecast, denoted by $\widehat{y}^{\star}_{k+1}$, can be obtained by solving the following optimization problem:

$$\widehat{y}^{\star}_{k+1}(\mathcal{F}_k) = \arg\min_{\widehat{y}_{k+1}} \mathcal{R}(\widehat{y}_{k+1}; \mathcal{F}_k) = \arg\min_{\widehat{y}_{k+1}} \int_{-\infty}^{\infty} \ell(y, \widehat{y}_{k+1})\, f_{Y_{k+1}|\mathcal{F}_k}(y \mid \mathcal{F}_k)\, dy. \tag{2}$$

This optimization program identifies the value of $\widehat{y}_{k+1}$ that best balances the conditional expected errors according to the loss function and the forecast distribution. The approach systematically determines the optimal predictor, ensuring alignment with the probabilistic structure of the data.

Two notable loss functions lead to optimal forecasts that have interpretable closed-form expressions:

- **Squared Error Loss:** When the loss function is the squared error, $\ell(y, \widehat{y}_{k+1}) = (y - \widehat{y}_{k+1})^2$, the optimal forecast is the conditional mean of the forecast distribution:
$$\widehat{y}^{\star}_{k+1}(\mathcal{F}_k) = \mathbb{E}[Y_{k+1} \mid \mathcal{F}_k].$$
- **Absolute Error Loss:** When the loss function is the absolute error, $\ell(y, \widehat{y}_{k+1}) = |y - \widehat{y}_{k+1}|$, the optimal forecast is the conditional median of the forecast distribution:
$$\widehat{y}^{\star}_{k+1}(\mathcal{F}_k) = \text{median}(Y_{k+1} \mid \mathcal{F}_k).$$

These specific loss functions offer direct interpretations: the squared error emphasizes the average outcome, while the absolute error focuses on the median outcome, illustrating how different loss functions tailor forecasts to different aspects of the forecast distribution.

2 Data-Driven Forecasting Approaches

In practical forecasting, the true forecasting density function $f_{Y_{k+1}|\mathcal{F}_k}$ is generally unknown, making the analytical solution of the problem of minimizing the total loss in (2) infeasible. As a result, forecasts must be derived by approximating the true distribution using available empirical data, typically presented as a finite sample path of length L, denoted by $y_{1:L} = (y_1, \ldots, y_L)$. The primary challenge, therefore, lies in effectively leveraging observed data to construct accurate forecast estimates.

In the following subsections, we explore several approaches to data-driven forecasting, each offering distinct perspectives on modeling and predicting future values:

1. **Model-Based:** This method assumes a specific model for the data-generating process, allowing us to exploit the structure of the underlying system to produce forecasts.

2. **Bayesian:** Building upon the model-based framework, the Bayesian approach incorporates uncertainty by treating model parameters as random variables, providing a principled mechanism for updating beliefs about the data as new information becomes available.

3. **Traditional Machine Learning:** This approach assumes a parametric form for the predictor, employing machine learning algorithms to learn the relationship between inputs and the next step in the time series.

4. **Deep Learning:** Leveraging the capabilities of neural networks, these methods capture complex, non-linear patterns in data, making them particularly effective for high-dimensional and challenging forecasting tasks.

2.1 Model-Based Approach

The model-based approach assumes a predefined structure for the data-generating process, typically represented by linear stochastic models such as autoregressive processes, moving average models, and their combinations. These models provide a systematic framework for understanding and forecasting time series data by capturing dependencies between current and past values of the series.

For instance, the AR model of order 1 assumes that the data is generated according to a specific parameterized recursion given by:

$$Y_{k+1} = \phi Y_k + \epsilon_k, \quad \text{where } \epsilon_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2) \text{ and } Y_0 = y_0.$$

We denote the vector of model parameters by $\boldsymbol{\theta}$. In the case of the AR(1) model, this vector is given by $\boldsymbol{\theta} = \begin{bmatrix} \phi & \sigma_\epsilon^2 & y_0 \end{bmatrix}^\intercal$. It is important to note that the initial condition y_0 should be included in the parameter vector when the process is not

in a stationary regime. In contrast, the initial condition can be omitted when the time-series data is measured during a stationary regime.

The chosen model inherently imposes a specific parametric form on the (unknown) one-step-ahead forecast density $f_{Y_{k+1}|\mathcal{F}_k}$. For instance, in the case of the AR(1) process operating in the stationary regime, the forecast density is Gaussian with mean $\phi \cdot y_k$ and variance σ_ϵ^2, i.e.,

$$f_{Y_{k+1}|\mathcal{F}_k}(\widehat{y}_{k+1} \mid Y_{1:k} = y_{1:k}) = f_{\boldsymbol{\theta}}(\widehat{y}_{k+1}; y_{1:k}) = \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(\widehat{y}_{k+1} - \phi y_k)^2}{2\sigma_\epsilon^2}\right).$$

This parametric version of the one-step-ahead density function, $f_{\boldsymbol{\theta}}(\widehat{y}_{k+1}; y_{1:k})$, directly reflects the structure of the AR(1) model and provides a probabilistic description of the forecast, capturing the uncertainty inherent in the process. It is important to note that the parametric function $f_{\boldsymbol{\theta}}$ approximates the true forecast density well only when the assumptions underlying the model closely match the actual data-generating process. The quality and reliability of the forecast density depend critically on the alignment between the model's structural assumptions and the true nature of the data.

2.1.1 Maximum Likelihood Estimation (MLE)

Once an appropriate model has been selected, its parameters $\boldsymbol{\theta}$ are typically estimated using **Maximum Likelihood Estimation (MLE)**, which identifies the parameter values most likely to have generated the observed data. Given a set of observations $y_{1:L} = (y_1, y_2, \ldots, y_L)$, the joint probability density of the observed data, conditioned on the initial conditions, which we characterize using the information set $\mathcal{F}_0$, is given by $f_{Y_{1:L}|\mathcal{F}_0}(Y_{1:L} \mid \mathcal{F}_0)$.

Unlike tabular data with i.i.d. samples, time series data exhibit strong temporal dependencies between observations. To properly compute the likelihood function for time series, we can employ the following *causal factorization* conditioned on the initial conditions:

$$f_{Y_{1:L}|\mathcal{F}_0}(Y_{1:L} \mid \mathcal{F}_0) = \prod_{k=0}^{L-1} f_{Y_{k+1}|\mathcal{F}_k}(y_{k+1} \mid \mathcal{F}_k), \tag{3}$$

where $\mathcal{F}_0$ contains all information about the initial conditions. Assuming that $f_{\boldsymbol{\theta}}$ provides a realistic parametric form for the true forecast density $f_{Y_k|\mathcal{F}_{k-1}}$, we can substitute the parametric expression $f_{\boldsymbol{\theta}}$ for the unknown one-step-ahead density $f_{Y_{k+1}|\mathcal{F}_k}$ in the causal factorization in (3) to derive the likelihood function:

$$\mathscr{L}(\boldsymbol{\theta}; y_{1:L}) = \prod_{k=0}^{L-1} f_{\boldsymbol{\theta}}(y_{k+1}; y_{1:k}).$$

This likelihood function measures the plausibility of the observed data $y_{1:L}$ as a function of the parameters of the model.

The MLE criterion seeks to find the parameter values $\boldsymbol{\theta}^\star$ that maximize the likelihood function given the observed data, thereby identifying the set of parameters that make the observed data most plausible. Mathematically, the MLE estimation problem is formulated as:

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \mathscr{L}(\boldsymbol{\theta}; y_{1:L}).$$

This optimization provides a systematic way to infer the model parameters that best capture the underlying data-generating process. By maximizing the likelihood, MLE ensures that the fitted model aligns closely with the observed data, making it a cornerstone technique in time series analysis.

Example 1: MLE Estimation for an AR(1) Model

Suppose that we assume the following AR(1) process as our model-generating process:

$$Y_{k+1} = \phi Y_k + \epsilon_k, \quad \text{where } \epsilon_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2).$$

We assume that our observations are taken in the stationary regime, meaning that the effect of the initial conditions has become negligible. Therefore, the vector of parameters can be simplified to $\boldsymbol{\theta} = \begin{bmatrix}\phi & \sigma_\epsilon^2\end{bmatrix}^\intercal$.
In the AR(1) model, the parametric form of the one-step-ahead forecast density is:

$$f_{\boldsymbol{\theta}}(\widehat{y}_{k+1}; y_{1:k}) = \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(y_{k+1} - \phi y_k)^2}{2\sigma_\epsilon^2}\right). \tag{4}$$

The likelihood function for the observed data $y_{1:L}$ is given by the causal factorization:

$$\mathscr{L}(\phi, \sigma_\epsilon^2; y_{1:L}) = \prod_{k=1}^{L-1} f_{\boldsymbol{\theta}}(y_{k+1}; y_{1:k}) = \prod_{k=1}^{L-1} \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(y_{k+1} - \phi y_k)^2}{2\sigma_\epsilon^2}\right).$$

This likelihood function encapsulates the plausibility of observing the data sequence $y_{1:k}$ for specific values of the parameters in $\boldsymbol{\theta}$.
The estimation of $\boldsymbol{\theta}$ involves maximizing the likelihood function with respect to the parameters ϕ and σ_ϵ^2. This is achieved by differentiating the likelihood function with respect to each parameter and setting the derivatives to zero[a]. Solving the resulting system of equations, we obtain the optimal MLE estimates as functions of the observed data $y_{1:L}$:

$$\phi^\star = \frac{\sum_{k=1}^{L-1} y_{k+1} y_k}{\sum_{k=1}^{L-1} y_k^2}, \quad \sigma_\epsilon^{2\star} = \frac{1}{L-1} \sum_{k=1}^{L-1} (y_{k+1} - \phi^\star y_k)^2.$$

These estimates represent the parameter values that maximize the likelihood of the observed data under the AR(1) model, providing the most plausible explanation of the time series dynamics within the framework of the assumed model structure.

[a] To simplify the differentiation process, a logarithmic transformation is commonly applied to the likelihood function, converting products into sums and exponentials into linear terms, which facilitates the calculation of the derivatives.

In conclusion, the strength of the model-based approach lies in its mathematical rigor and interpretability, particularly when the model is a simple linear one. Linear models are grounded in well-established statistical theory, providing clear and transparent relationships between variables that make them easy to understand and validate. This interpretability makes the linear-model-based approach particularly appealing in fields where understanding the underlying dynamics and communicating results clearly are crucial.

2.2 Bayesian Approach

The Bayesian approach extends the model-based approach by incorporating prior knowledge or beliefs about the model parameters through prior distributions. Unlike classical model-based methods, which treat parameters as fixed but unknown quantities, the Bayesian framework treats parameters $\boldsymbol{\theta}$ as random variables with their own probability distributions. This perspective allows for the explicit incorporation of uncertainty about the parameters into the model, making it a powerful tool for inference and forecasting in complex and uncertain environments.

2.2.1 Sequential Bayesian Updating

In the Bayesian framework, our understanding of the parameters at time k is initially represented by a **prior density** $f_{\boldsymbol{\theta}|\mathcal{F}_k}$, which encodes existing beliefs or expert knowledge about the parameters based on all available information up to time k. As new data becomes available at time $k+1$, Bayes' rule is used to update the prior beliefs, resulting in the **posterior density** $f_{\boldsymbol{\theta}|\mathcal{F}_{k+1}}$. The posterior density combines the likelihood of the new observation with the prior density, providing an updated view of the parameters' density. Mathematically, the posterior density is computed

according to Bayes' rule, as follows:

$$\underbrace{f_{\boldsymbol{\theta}|\mathcal{F}_{k+1}}}_{\text{Posterior}} = \frac{\overbrace{f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{F}_k}}^{\text{Likelihood}} \cdot \overbrace{f_{\boldsymbol{\theta}|\mathcal{F}_k}}^{\text{Prior}}}{\underbrace{f_{Y_{k+1}|\mathcal{F}_k}}_{\text{Marginal Likelihood}}}.$$

This equation contains the following elements:

- The **prior density** at time k, denoted by $f_{\boldsymbol{\theta}|\mathcal{F}_k}$, which represents our beliefs about the probability density of the parameters before observing Y_{k+1}.
- The **likelihood function**, denoted by $f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{F}_k}$, which measures the likelihood of the new observation $Y_{k+1} = y_{k+1}$ given specific parameter values $\boldsymbol{\theta}$ and previous information $\mathcal{F}_k$.
- The **marginal likelihood**, denoted by $f_{Y_{k+1}|\mathcal{F}_k}$, normalizes the posterior with respect to the parameters $\boldsymbol{\theta}$. It is computed as:

  $$f_{Y_{k+1}|\mathcal{F}_k} = \int f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{F}_k} \cdot f_{\boldsymbol{\theta}|\mathcal{F}_k} \, d\boldsymbol{\theta}.$$

  This term represents the total probability of observing the data under the proposed model, integrating over all possible parameter values weighted by their prior distributions. It effectively quantifies the overall support that the data provides for the model in its entirety, independent of any particular parameter configuration. Commonly referred to as **evidence**, this quantity assesses the plausibility that the observed data was generated by the specified model and is an important metric in Bayesian model selection.

The resulting posterior density $f_{\boldsymbol{\theta}|\mathcal{F}_{k+1}}$ encapsulates all updated information about the parameters, blending prior beliefs with the empirical evidence from the latest observation $Y_{k+1} = y_{k+1}$. This process naturally accommodates parameter uncertainty by continuously updating the belief about the parameters as new data becomes available. Unlike other approaches, which provide single-point estimates and often ignore parameter variability, the Bayesian framework allows the inference to reflect a full distribution of plausible parameter values. This richer representation captures the inherent uncertainty and variability in the parameters, offering a comprehensive understanding of how the data informs the model, rather than reducing it to a single, potentially misleading, point estimate.

Example 2: Bayesian Approach for the AR(1) Model

Consider an AR(1) model operating in the stationary regime. Suppose that at the current time k, our beliefs about the parameters are informed by the observed time series $y_{1:k}$. We aim to update these beliefs regarding the parameters $\boldsymbol{\theta} = \begin{bmatrix}\phi & \sigma_\epsilon^2\end{bmatrix}^\intercal$ through Bayesian inference, using a new observation $Y_{k+1} = y_{k+1}$. Prior to observing the new data point y_{k+1}, our **prior beliefs** about the parameters ϕ and σ_ϵ^2 are represented by the following densities:

1. **Prior for the parameter** ϕ: We assume that the prior density $f_{\phi|\mathcal{F}_k}(\phi \mid y_{1:k})$ follows a normal distribution with mean μ_ϕ and variance σ_ϕ^2, where the values of these hyperparameters are selected based on prior knowledge or expert judgment. This density captures our uncertainty about the value of the parameter ϕ at time k.

2. **Prior for the parameter** σ_ϵ^2: For the variance parameter, we adopt an inverse-gamma prior[a], i.e., $\sigma_\epsilon^2 \sim \text{Inv-Gamma}(\alpha, \beta)$, where the shape parameter α and the scale parameter β are determined by prior experience or domain expertise.

The **likelihood function** quantifies the plausibility of observing the new data point y_{k+1} given the current parameter values and past observations. It serves as a measure of how well the parameters ϕ and σ_ϵ^2 account for the new observation y_{k+1} in light of the previously observed data. For the AR(1) model, the likelihood of the new observation given the parameters ϕ, σ_ϵ^2, and the historical data $y_{1:k}$ is expressed as:

$$f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{Y}_k}(y_{k+1} \mid \phi, \sigma_\epsilon^2, y_{1:k}) = \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(y_{k+1} - \phi y_k)^2}{2\sigma_\epsilon^2}\right).$$

Applying Bayes' rule, we combine the prior densities with the likelihood function to derive the updated posterior density of the parameters given the new observation y_{k+1}:

$$\overbrace{f_{\boldsymbol{\theta}|\mathcal{F}_{k+1}}(\phi, \sigma_\epsilon^2 \mid y_{1:k+1})}^{\text{Updated Posterior}} = \frac{\overbrace{f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{F}_k}(y_{k+1} \mid \phi, \sigma_\epsilon^2, y_{1:k})}^{\text{Likelihood}} \cdot \overbrace{f_{\phi|\mathcal{F}_k} \cdot f_{\sigma_\epsilon^2|\mathcal{F}_k}}^{\text{Priors}}}{\underbrace{f_{Y_{k+1}|\mathcal{F}_k}(y_{k+1} \mid y_{1:k})}_{\text{Marginal Likelihood}}}.$$

The **marginal likelihood** in the denominator ensures that the posterior density is properly normalized. However, this term often poses computational challenges due to the high-dimensional integration required. Notably, in practical Bayesian inference, the marginal likelihood is unnecessary for parameter

estimation since it does not depend on the parameters ϕ and σ_ϵ^2. Consequently, Bayesian updating can proceed using the unnormalized posterior density, bypassing explicit calculation of the marginal likelihood. Expanding the numerator of the **updated posterior**, the expression for the **unnormalized posterior density** is given by:

$$f_{\boldsymbol{\theta}|\mathcal{F}_{k+1}}(\phi, \sigma_\epsilon^2 \mid y_{1:k+1}) \propto \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(y_{k+1} - \phi y_k)^2}{2\sigma_\epsilon^2}\right) \cdot f_{\phi|\mathcal{F}_k} \cdot f_{\sigma_\epsilon^2|\mathcal{F}_k}.$$

This posterior density reflects the updated beliefs about the parameters by coherently integrating prior information with the empirical evidence provided by the latest observation.

[a] The inverse-gamma distribution is defined as $f(x) \propto x^{-(\alpha+1)} \exp(-\beta/x)$ for $x > 0$.

2.2.2 Markov Chain Monte Carlo Sampling

Deriving the updated posterior density analytically is often infeasible, particularly when the posterior lacks a closed-form solution due to the complexity of the underlying model and the prior densities. In such scenarios, numerical methods, such as **Markov Chain Monte Carlo (MCMC)**, are employed to approximate the posterior distribution by generating a collection of samples. MCMC is especially valuable in Bayesian inference as it allows us to explore complex posterior landscapes that are otherwise computationally intractable. Sampling from the posterior is crucial because, even without a closed-form expression, these samples allow us to estimate essential properties of the distribution, such as empirical means, variances, and credible intervals, providing insights into parameter uncertainties and the overall model behavior.

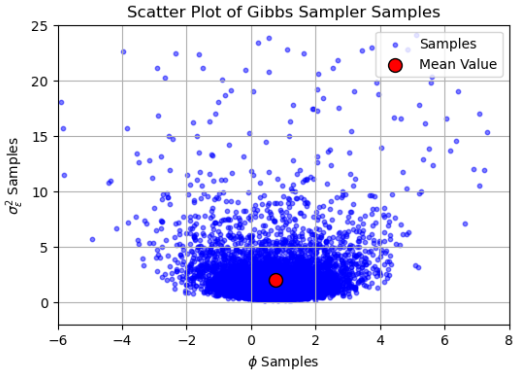

Figure 1: Samples drawn from a posterior distribution using the Gibbs Sampler.

Two prominent MCMC techniques are the **Metropolis-Hastings** algorithm and the **Gibbs sampler**. In Python, these methods can be implemented using libraries such as `NumPy` and `SciPy`, which provide efficient tools for sampling from commonly used distributions, such as the normal and inverse-gamma. Figure 1 illustrates the samples obtained using the Gibbs sampler on the posterior for an AR(1) model with a Gaussian prior on ϕ and an inverse-gamma prior on σ^2.

As more data is observed over time, we can sequentially update the posterior by treating the posterior at time k as the prior for the subsequent time step $k+1$, applying Bayes' rule:

$$f_{\boldsymbol{\theta}|\mathcal{F}_{k+1}} \propto f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{F}_k} \cdot f_{\boldsymbol{\theta}|\mathcal{F}_k}.$$

This expression shows that the updated posterior at time $k+1$ is proportional to the likelihood of the new observation $Y_{k+1} = y_{k+1}$ multiplied by the posterior from the previous step. When deriving an analytical expression is infeasible, we approximate the new posterior using a cloud of samples from the previous posterior, weighted according to the likelihood of the new data. This iterative approach allows the model to evolve dynamically, continuously refining its parameter estimates as new data becomes available. Figure 2 demonstrates the evolution of the sample cloud generated by the Gibbs sampler for increasing time steps, illustrating the concentration of the posterior distribution as more data is incorporated. As the number of observations grows, the posterior becomes more concentrated, reflecting increased certainty in the parameter estimates and the model's adaptation to the cumulative evidence from the data stream.

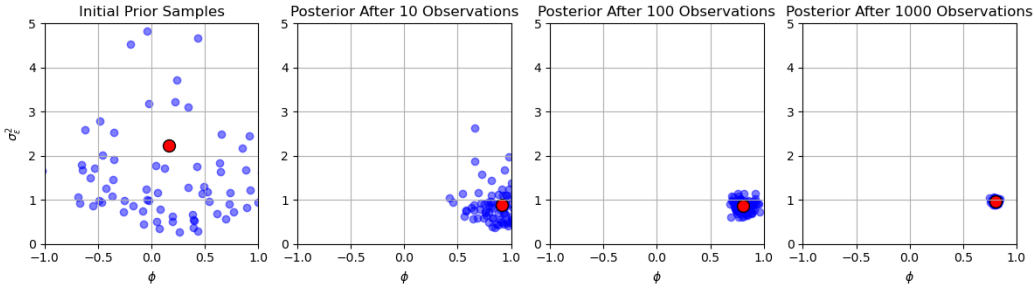

Figure 2: Posterior samples after 1, 10, 100, and 1000 updates, showing the concentration of the posterior as the number of observations increases.

Include connection with Markov Chains in next version... Theoretical and implementation details about Metropolis-Hasting and Gibbs Sampling...

2.2.3 Bayesian Forecasting

Bayesian forecasting offers a rigorous framework for incorporating parameter uncertainty directly into the prediction of future values. Unlike traditional approaches

that yield single-point forecasts, the Bayesian methodology generates a predictive distribution that encompasses the full range of potential future outcomes, weighted by their likelihood given the observed data. The one-step-ahead forecast density for the next observation Y_{k+1}, conditioned on the information available up to time k, can be computed using the posterior and prior densities at time k through the following integration:

$$f_{Y_{k+1}|\mathcal{F}_k} = \int f_{Y_{k+1}|\boldsymbol{\theta},\mathcal{F}_k} \cdot f_{\boldsymbol{\theta}|\mathcal{F}_k} \, d\boldsymbol{\theta}.$$

This expression reflects the Bayesian principle of integrating over the entire posterior distribution of the parameters, effectively averaging predictions across all plausible parameter values weighted by their posterior probabilities. Consequently, the resulting predictive distribution accounts for both the inherent variability in the data and the uncertainty surrounding the parameter estimates. However, due to the complexity of most models, this integral is often analytically intractable. In practice, it is approximated using numerical techniques such as Monte Carlo Integration or Variational Inference, which provide feasible solutions to evaluate the one-step-ahead forecast density.

In conclusion, the Bayesian approach offers a powerful and adaptable framework for statistical modeling and forecasting by coherently integrating prior knowledge with observed data. It provides a principled way to quantify uncertainty, making it especially valuable in settings that require a nuanced understanding of risk. While the computational demands of Bayesian methods necessitate sophisticated techniques to render them practicable, the insights gained from this approach often surpass those achievable through traditional methods, affirming its central role in modern statistical inference.

2.3 Traditional Machine Learning Approach

In model-based forecasting, traditional time series models such as Autoregressive (AR), Moving Average (MA), and their combinations (ARMA) rely on predefined parameterized structures to describe the data-generating process. While these models have well-established theoretical foundations and are effective for many stationary time series, their inherent rigidity can become a significant limitation when the true dynamics of the data deviate from the underlying assumptions. Real-world data often exhibit complexities such as non-linearity, non-stationarity, seasonality, and heavy-tailed noise, which cannot be adequately captured by simple parameterized models. As a result, the use of these models in complex settings can lead to biased forecasts, underestimated uncertainty, and misleading inferences.

Machine learning approaches have emerged as powerful alternatives due to their ability to learn complex patterns directly from the data. Unlike traditional model-based methods, machine learning models do not require the specification of the

full data-generating process. Instead, they focus on learning the direct mapping between input features extracted from past observations and the target forecast variable $\widehat{y}_{k+1}$. Machine learning techniques are able to capture non-linear relationships, adapt to evolving trends, and model intricate dependencies that may be overlooked by classical methods. By leveraging large datasets and advanced training algorithms, machine learning techniques provide effective forecasting solutions that are particularly valuable when the underlying data generating process is too complex.

In machine learning-based forecasting, the goal is to learn a parameterized function $\Phi_{\boldsymbol{\theta}}$ that maps a vector of input features at time k, denoted by $\mathbf{x}_k$, directly into the forecasted value $\widehat{y}_{k+1}$:

$$\widehat{y}_{k+1} = \Phi_{\boldsymbol{\theta}}(\mathbf{x}_k),$$

where $\boldsymbol{\theta}$ denotes the model parameters to be learned from the data. This function $\Phi_{\boldsymbol{\theta}}$ can be constructed by various algorithms, including regression trees, random forests, gradient boosting machines, and support vector regression (SVR). These techniques do not impose restrictive assumptions on the functional form of the data-generating process, enabling them to adapt to complex, non-linear relationships and evolving patterns that traditional models may fail to capture. By leveraging large datasets and advanced computational techniques, these algorithms can effectively learn the mapping from historical data to future predictions, providing a flexible alternative to classical time series models in handling real-world data complexities.

2.3.1 Feature Engineering for Temporal Data

Constructing a vector of informative input features $\mathbf{x}_k$ is crucial in time series learning, as it directly influences the model's ability to capture temporal dependencies. For time series forecasting, designing these input features $\mathbf{x}_k$ requires careful consideration of the sequential nature of the data. The input features $\mathbf{x}_k$ can only depend on past information contained in the information set $\mathcal{F}_k$, which includes all observations and relevant data available up to time k. This constraint ensures that the predictor does not inadvertently use future information, preserving the causality required for valid forecasting.

Traditional machine learning approaches typically rely on human-designed features, a process known as **feature engineering**, where domain knowledge and exploratory data analysis guide the selection, transformation, and creation of features believed to capture the most relevant aspects of the data. Typical features used in time series forecasting include:

- **Lagged Values**: Past observations of the series, such as $y_k, y_{k-1}, \ldots, y_{k-p}$, where p is the number of lagged terms considered.
- **Rolling Statistics**: Moving averages, rolling standard deviations, and other

windowed statistical measures that summarize recent behavior.

- **Seasonality Indicators**: Variables indicating the time of day, day of the week, month, or season, capturing periodic patterns.
- **Exogenous Variables**: External factors or covariates that may influence the target variable, such as economic indicators, weather data, or market indices.
- **Time Series Transformations**: Techniques like differencing to remove trends or scaling to normalize seasonal effects, enhancing model performance and stability.

2.3.2 Model Training with Temporal Dependencies

Unlike tabular i.i.d. data, time series data cannot be randomly split into training, validation, and test sets without violating the inherent temporal structure. Random shuffling of data would disrupt the sequential order and could lead to information leakage, where future information inadvertently influences model training. To preserve the causality and temporal dependencies of the data, it is essential to partition the data *sequentially*, ensuring that the training set precedes the validation and test sets. This approach respects the natural order of observations, preventing the inadvertent incorporation of future data during the training process.

Machine learning methods differ fundamentally from model-based approaches, which employ stochastic models to implicitly define a parameterized form for the one-step-ahead forecast distribution $f_{\boldsymbol{\theta}}$. Instead, machine learning techniques aim to directly approximate the solution to the risk minimization problem specified in (2) by training the parameterized function $\Phi_{\boldsymbol{\theta}}$. This function utilizes a feature vector $\mathbf{x}_k$, extracted from the information set $\mathcal{F}_k$, to approximate the optimal predictor $\widehat{y}^{\star}_{k+1}(\mathcal{F}_k)$.

The raw training data consists of a finite sequence of observations $y_{1:L}$, which is preprocessed to construct a training set of input-output pairs $\{(\mathbf{x}_k, y_{k+1})\}_{k=1}^{L-1}$. Each input vector $\mathbf{x}_k$ is designed based on domain knowledge, as discussed in the preceding subsection, and encapsulates relevant information from $\mathcal{F}_k$. The training objective is to determine the set of parameters $\boldsymbol{\theta}$ that minimizes the **total loss**, defined as:

$$\mathcal{L}(\boldsymbol{\theta}; y_{1:L}) = \frac{1}{L-1} \sum_{k=0}^{L-1} \ell\left(y_{k+1}, \Phi_{\boldsymbol{\theta}}(\mathbf{x}_k)\right),$$

where $\Phi_{\boldsymbol{\theta}}(\mathbf{x}_k)$ represents the parameterized mapping of input features $\mathbf{x}_k$ to the forecasted value $\widehat{y}_{k+1}$. The total loss $\mathcal{L}(\boldsymbol{\theta})$ measures the average discrepancy between the observed values y_{k+1} and the predicted values $\widehat{y}_{k+1}$ across all training steps. Optimization algorithms such as **gradient descent** (and its variants) are typically employed to minimize the total loss. These algorithms ensure that $\Phi_{\boldsymbol{\theta}}$ captures the

essential patterns and dependencies within the time series, ultimately improving its forecasting accuracy and performance.

2.3.3 Validation of Forecasting Machine Learning Techniques

Evaluating machine learning models for time series forecasting differs significantly from evaluating models with independent and identically distributed (i.i.d.) samples. Standard cross-validation techniques that randomly shuffle data cannot be directly applied because they would disrupt the sequential order of observations, leading to information leakage. Instead, time series-specific evaluation methods are used:

1. **Rolling Forecast Origin (Walk-Forward Validation)**: The data is split into a series of expanding training and test sets, where the model is trained on past data and tested on future observations in a sequential manner. This approach preserves the temporal structure and mimics real-time forecasting scenarios.

2. **Blocked Cross-Validation**: This method involves splitting the data into contiguous blocks to ensure that the validation set always follows the training set temporally, thus maintaining the natural order of the observations.

Machine learning approaches offer substantial advantages in forecasting due to their flexibility, adaptability, and ability to capture complex relationships in data. However, these methods also come with challenges, including the need for large datasets, the risk of overfitting, and the requirement for careful hyperparameter tuning. Additionally, the black-box nature of many machine learning models can complicate interpretability, which is critical in applications where understanding the driving factors behind forecasts is essential. A more in-depth coverage of these techniques, including strategies to address their inherent challenges, will be provided in Chapter ?, where we explore the theoretical foundations, practical implementation, and interpretability solutions for advanced machine learning models in time series forecasting.

2.4 Deep Learning Approaches for Time Series Forecasting

Deep learning provides us with a powerful framework for time series forecasting, capable of modeling complex, non-linear relationships and capturing long-term dependencies that traditional statistical and machine learning methods often struggle with. Unlike traditional approaches, which rely on human-crafted features or predefined structures, deep learning models learn hierarchical representations directly from raw data. In these models, early processing layers typically capture simpler patterns, such as short-term trends or local variations, while later layers learn more abstract, complex patterns, such as long-term dependencies or interactions between

variables. This multi-level processing makes deep learning highly effective in dealing with intricate patterns, non-stationarities, and high-dimensional inputs.

One of the primary strengths of deep learning in time series forecasting lies in its ability to automatically learn temporal dependencies, which are critical for accurate predictions. Below, we outline the main families of deep learning techniques commonly used for time series forecasting:

TCN->RNN or RNN->TCN?

1. **Recurrent Neural Architectures**: Including RNNs, LSTMs, GRUs, and Neural ODEs, these models maintain an internal state to capture both short- and long-term dependencies in sequential data.
2. **Convolutional Neural Architectures**: Models like TCNs, Convolutional LSTMs, and Convolutional GNNs use convolutional filters to extract temporal features and efficiently model hierarchical patterns while respecting causality.
3. **Attention Mechanisms and Transformers**: These models utilize self-attention to capture long-range dependencies without relying on sequential processing, improving over traditional recurrent models.
4. **Autoencoders**: Autoencoders, including VAEs and Seq2Seq frameworks, compress time series into latent spaces, enabling multi-step forecasting and reconstruction.
5. **State-Space Models**: State-Space Models, such as Mamba, describe latent state evolution over time, offering a probabilistic approach to capturing temporal dynamics and uncertainty.

We elaborate on each of these techniques in the subsections below.

2.4.1 Recurrent Neural Architectures

Recurrent Neural Networks (RNNs) are a foundational class of deep learning models specifically designed to handle sequential data, such as time series. Unlike traditional feedforward neural networks, RNNs maintain an **internal state** that evolves recursively over time, functioning as a dynamic memory. This internal state efficiently compresses and retains relevant information from previous time steps. At each time step, the internal state is updated based on both the current input and the preceding state, thereby enabling RNNs to capture and model temporal dependencies. The final output of the RNN at each time step is generated through an **observation equation**, which transforms the current internal state into an observable prediction of the subsequent time step.

One of the main strengths of RNNs is their ability to capture temporal dependencies in the data. Furthermore, RNNs are flexible in their application to sequences of

varying lengths, making them highly adaptable across different forecasting horizons. Additionally, unlike traditional forecasting models, which require manual feature engineering, RNNs automatically learn representations from raw data, reducing the need for extensive preprocessing or the crafting of features.

Despite these strengths, traditional RNNs present its limitations. One of the most significant challenges they face is the **vanishing gradient problem**, which arises during the training process. This issue makes it difficult for the network to learn long-term dependencies, as information from earlier time steps cannot effectively influence later stages of the sequence. Consequently, RNNs are often limited in their ability to retain relevant information over extended sequences. This can be particularly problematic for time series data with long-term dependencies, where the relationships between observations may span significant time intervals.

To overcome these limitations, more advanced variants of RNNs have been developed, including **Long Short-Term Memory** (LSTM) networks and **Gated Recurrent Units** (GRU). These architectures introduce gating mechanisms that regulate the flow of information, enabling the network to retain or discard information more effectively. By addressing the vanishing gradient problem and enhancing memory retention, LSTMs and GRUs improve the model?s ability to capture long-term dependencies in time series data with greater accuracy and stability. In Chapter ?, we will discuss these architectures in detail, along with their applications and limitations..

2.4.2 Convolutional Neural Architectures

Convolutional Neural Networks (CNNs) were originally developed for image processing tasks, where they excel at capturing spatial hierarchies through convolutional operations. In the context of images, the convolution operation applies a filter (or kernel) across various regions to detect local patterns, such as edges or textures, which are then progressively aggregated across layers to form increasingly complex features.

Although CNNs were initially applied in the spatial domain, *convolutional filters* can be used to process temporal data, enabling the network to capture local temporal dependencies in time series. In this setting, the convolutional filter slides is able to effectively detect short-term patterns, such as trends or seasonal behaviors, in the time series data. To capture longer-term dependencies beyond the immediate neighborhood of each time step, *dilated convolutions* are employed. Dilated convolutions apply the filter to inputs spaced by gaps (or dilations), allowing the model to efficiently process a wider range of past data without increasing the model's depth or the number of parameters.

A key advantage of CNNs in time series forecasting lies in their capacity for parallel processing. Unlike Recurrent Neural Networks (RNNs), which operate se-

quentially on input data, CNNs can apply filters to different segments of a sequence simultaneously, significantly improving computational efficiency, especially for long sequences. Moreover, the hierarchical nature of CNNs—where multiple layers of convolutions capture both local and global temporal patterns—allows them to learn sophisticated temporal representations, analogous to how they capture spatial hierarchies in images.

In Chapter ?, we will examine the foundational principles of causal convolutions and CNNs. Additionally, we will explore specific adaptations, such as Temporal Convolutional Networks (TCNs), Convolutional LSTMs, WaveNet, and Convolutional Graph Neural Networks (GNNs), which extend the basic ideas of convolutional operations and enhance their effectiveness through innovations like dilations and gated mechanisms.

2.4.3 Attention Mechanisms and Transformers

Attention mechanisms and Transformer models represented a significant advancement in sequential data modeling by focusing on the relevance of different time steps rather than processing data sequentially. Transformers replace the recurrent processing found in RNNs with parallel processing and self-attention, enabling the model to weigh the importance of various parts of the input sequence independently. The self-attention mechanism in Transformers assigns different weights to different time steps, allowing the model to focus on the most relevant information for forecasting. This approach eliminates the need for sequential processing, making Transformers highly efficient and scalable, particularly for large datasets and complex forecasting tasks. Transformers excel at capturing long-range dependencies and complex relationships within the data, offering a flexible framework that adapts to varying data structures and patterns.

Attention mechanisms are also integrated into other deep learning architectures, enhancing their ability to selectively capture relevant temporal information. This adaptability makes attention-based models versatile and highly effective for time series forecasting, particularly when long-term dependencies or intricate temporal patterns are present. In Chapter ?, we will explore these attention mechanisms in depth, covering their integration into various architectures, their applications in time series forecasting, and the challenges they address.

2.4.4 Autoencoders and Sequence-to-Sequence

Autoencoder-based architectures, including Sequence-to-Sequence models and Variational Autoencoders (VAEs), provide a framework for learning latent representations of time series data. Sequence-to-Sequence (Seq2Seq) models use an encoder-decoder structure to map input sequences to output sequences, allowing the model to learn a compressed representation of the input that captures the temporal depen-

dencies. This architecture is particularly useful for multi-step forecasting scenarios, where the input and output sequences may differ in length.

VAEs extend the autoencoder framework by introducing a probabilistic approach, where the encoder learns to map the input data into a latent space that captures the underlying distribution of the time series. The decoder then reconstructs the time series from this latent space, allowing the model to capture complex temporal structures and variability in the data. VAEs are particularly useful when uncertainty quantification is important, as they provide a probabilistic interpretation of the learned representations.

Both Seq2Seq models and VAEs emphasize the ability to learn efficient representations that encapsulate the temporal structure and variability of the data, offering a flexible approach to modeling complex time series. In Chapter ?, we will delve into these architectures, exploring their mechanisms for capturing temporal dependencies, their strengths in handling multi-step forecasting, and their practical applications in time series analysis.

2.4.5 State-Space Models

State-space models combine the principles of deep learning with probabilistic modeling to represent time series data through latent states and observed processes. Mamba, a state-space model, uses neural networks to parameterize the state-space representation, allowing the model to adaptively learn both the temporal dynamics and the observation processes. This approach provides a structured and flexible way to model sequential data, integrating the strengths of both neural and probabilistic frameworks. By treating the system as a combination of latent state evolution and noisy observations, state-space models offer a robust method for handling uncertainty and capturing complex temporal dependencies.

State-space models are particularly effective for handling non-stationary data, incorporating uncertainty, and modeling structural changes in the time series. By learning the underlying latent states, these models provide a principled way to capture the evolving dynamics of complex time series, making them a versatile tool for a wide range of forecasting applications. The latent states encode hidden aspects of the data that are not directly observed but inferred through the model, which allows for dynamic adaptation to changes in the time series structure, such as regime shifts or abrupt changes.

The state-space framework serves as a unified perspective that can accommodate many of the deep learning techniques discussed earlier. For instance, recurrent neural networks (RNNs) can be viewed as special cases of state-space models, where the hidden states evolve over time and influence the next observations. Similarly, sequence-to-sequence models and attention mechanisms can be integrated into a state-space representation, where the evolution of latent states and their influence

on observations are modeled explicitly. This unification provides a broader and more flexible framework for understanding and applying deep learning techniques to time series forecasting.

These deep learning techniques offer a comprehensive set of tools for addressing the challenges inherent in time series forecasting. By leveraging advanced neural architectures and learning directly from raw data, deep learning models provide a flexible and powerful alternative to traditional approaches, enabling accurate and robust forecasts even in the face of complex, non-linear, and evolving data patterns. An in-depth coverage of these techniques, including their theoretical foundations, implementation strategies, and practical applications, will be provided in the coming chapters, using the unified view provided by the state-space framework.

Exercises

1. (4 points) Consider the following Moving Average of order 1, MA(1), process:

$$Y_k = \epsilon_k + \theta\epsilon_{k-1}, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma_\epsilon^2),$$

where μ is a constant, and θ is the moving average coefficient.

Answer the following questions:

   (a) Derive the one-step-ahead forecast density $f_{Y_{k+1}|\mathcal{F}_k}(y_{k+1} \mid \mathcal{F}_k)$, where $\mathcal{F}_k = \{Y_k = y_k, Y_{k-1} = y_{k-1}, Y_{k-2} = y_{k-2}, \dots\}$. *Hint*: The information set does not contain the values of the noise terms.

   (b) Derive the value of the optimal forecast $\widehat{y}_{k+1}^\star$ when the loss function is the squared error.

   (c) Derive the value of the optimal forecast $\widehat{y}_{k+1}^\star$ when the loss function is the absolute error.

   (d) Derive the **two-steps-ahead forecast density**, $f_{Y_{k+2}|\mathcal{F}_k}(y_{k+2} \mid \mathcal{F}_k)$.

   (e) Derive the value of the two-steps-ahead optimal forecast $\widehat{y}_{k+2}^\star$ when the loss function is the squared error.

2. (6 points) Consider the following Autoregressive Moving Average process, denoted by ARMA(1,1) and defined as:

$$Y_k = \phi Y_{k-1} + \epsilon_k + \theta\epsilon_{k-1}, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2) \text{ and } Y_0 = 0,$$

where ϕ and θ are the autoregressive and moving average coefficients, respectively. Assume that, at time k, you have access to the following information set: $\mathcal{F}_k = \{Y_k = y_k, Y_{k-1} = y_{k-1}, Y_{k-2} = y_{k-2}, \dots\}$. Note that the information set does **not** contain the values of the noise terms ϵ_k.

Answer the following questions:

   (a) Derive the one-step-ahead forecast density $f_{Y_{k+1}|\mathcal{F}_k}(y_{k+1} \mid \mathcal{F}_k)$ for the ARMA(1,1) process.

   (b) Write down an explicit integral expression for the conditional risk for the squared error (SE) loss function. Your expression should be an integral where the integrand should be an explicit function of the parameters ϕ, θ, σ_ϵ^2, the prediction $\widehat{y}_{k+1}$, the values in the information set $\mathcal{F}_k$, and the integration variable y. *Hint*: The constant $\pi \approx 3.14$ should appear in your integral.

   (c) Use the properties of the Gaussian density to derive an explicit expression for the conditional risk for the SE loss function that does not involve integrals. The expression should be a function of the parameters ϕ, θ, σ_ϵ^2, the prediction $\widehat{y}_{k+1}$, and the values in the information set $\mathcal{F}_k$.

(d) Minimize the conditional risk function with respect to $\widehat{y}_{k+1}$ as follows: Find the derivative of the conditional risk with respect $\widehat{y}_{k+1}$, force the derivative to be zero, and find the value $\widehat{y}^{\star}_{k+1}$ that makes the derivative equal to zero.

(e) What is the relationship between this value and $\mathbb{E}[Y_{k+1} \mid \mathcal{F}_k]$? Derive an expression for this conditional expectation to verify your claim.

3. (4 points) Consider the following AR(1) model:

$$Y_k = \theta Y_{k-1} + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0,1) \text{ and } Y_0 = y_0.$$

where θ is the unknown parameter we want to estimate, and the noise variance is fixed at $\sigma_\epsilon^2 = 1$. You are provided with L observations of the time series $y_{1:L} = (y_1, y_2, \ldots, y_L)$.

Answer the following questions:

(a) Write down the likelihood function $\mathscr{L}(\theta; y_{0:L})$ for the AR(1) model using the causal factorization in (3).

(b) Derive the log-likelihood function, defined as $\log \mathscr{L}(\theta; y_1, y_2, \ldots, y_L)$, where $\log(\cdot)$ is the natural logarithm.

(c) Compute the derivative of the log-likelihood function with respect to θ.

(d) Set the derivative equal to zero and solve for θ to obtain the **Maximum Likelihood Estimate (MLE)** $\theta^\star$ as a function of the observations.

(e) Interpret the MLE expression in terms of the autocorrelation of the observed data.

4. (5 points) Consider the AR(1) model:

$$Y_k = \theta Y_{k-1} + \epsilon_k, \quad \text{with } \epsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0,1) \text{ and } Y_0 = y_0.$$

where θ is the unknown parameter we want to estimate, and the noise variance is fixed at $\sigma_\epsilon^2 = 1$. You are provided with L observations of the time series $y_{1:L} = (y_1, y_2, \ldots, y_L)$. Let us consider a prior normal prior distribution for the unknown parameter, i.e., $\theta \sim \mathcal{N}(\mu_0, \sigma_0^2)$, where μ_0 and σ_0 are given parameters.

Answer the following questions:

(a) Assume that at time $k = 1$, you observe a single data point $Y_1 = y_1$ from the AR(1) process. Derive both the likelihood function $f_{Y_1|\theta}$ and the posterior distribution $f_{\theta|Y_{0:1}}$ given this single observation y_1 (and the initial condition $Y_0 = y_0$. *Hint*: Show that the posterior distribution remains Gaussian and derive the updated posterior mean μ_1 and posterior

variance σ_L^1 after observing Y_1. In this task, you may find useful the following identity:

$$-\frac{1}{2}\left(\frac{(\theta-\mu_k)^2}{\sigma_k^2}+(y_{k+1}-\theta y_k)^2\right)=-\frac{(\theta-\mu_{k+1})^2}{2\sigma_{k+1}^2},$$

where $\sigma_{k+1}^2=\left(\frac{1}{\sigma_k^2}+y_k^2\right)^{-1}$ and $\mu_{k+1}=\sigma_1^2\left(\frac{\mu_k}{\sigma_k^2}+y_{k+1}y_k\right)$.

(b) Now assume you have observed L data points, $y_{1:L}=\{y_1,y_2,\ldots,y_L\}$. Derive a recursion to update the posterior mean and the posterior variance of the posterior distribution for θ as you incorporate more observations. What is the posterior density after incorporating all L observations. *Hint*: Show that the posterior distributions remain Gaussian and derive the updated posterior mean μ_L and posterior variance σ_L^2 after observing the entire sequence.

(c) Given the posterior distribution after L observations, derive the predictive distribution of the next value Y_{L+1} given the previous observation $Y_{1:L}=y_{1:L}$. *Hint*: Show that this predictive distribution is Gaussian and compute the predictive mean and variance. Compare the predictive mean with the Maximum Likelihood Estimate (MLE) $\theta^\star$ obtained in the previous exercise.

(d) Derive the predictive distribution for Y_{L+h} (i.e., the forecast h-steps ahead) given the data $y_{1:L}$ and the corresponding posterior distribution of θ. Compute the mean and variance of the h-step-ahead forecast distribution.

(e) Derive the 95% confidence intervals for the h-steps-ahead forecast distribution based on the predictive distribution for Y_{L+h} as h increases. Discuss how the width of the confidence intervals evolves with increasing h, and explain the factors contributing to this behavior.

5. Answer the following short questions (justify your answers):

(a) What is the most appropriate loss function when the one-step-ahead forecast distribution is heavy-tailed?

(b) What is the most appropriate loss function when the one-step-ahead forecast distribution is Gaussian?

(c) What are the key differences between the model-based approach and the Bayesian model-based approach?

(d) What are the advantages and disadvantages of the traditional machine learning approach compared to the model-based approach?

(e) What are the advantages and disadvantages of the deep learning approach compared to the traditional machine learning approach?

(f) What is the main limitation of Recurrent Neural Networks (RNNs)?

(g) In what scenarios are autoencoders particularly useful?

(h) How can long-term dependencies be incorporated into Convolutional Neural Networks (CNNs)?

(i) What are the advantages of Transformers compared to RNNs?

6. (From 1A—3 points) Consider a Markov chain $\mathcal{X} = \{X_1, X_2, \ldots\}$ with state space $\mathcal{S} = (1, 2, 3)$ and transition matrix:

$$P = \begin{bmatrix} 0 & 0.8 & * \\ 0.1 & * & 0.3 \\ 0.2 & 0.8 & * \end{bmatrix}.$$

(a) Fill in the unknown entries, denoted by $*$, in the transition matrix.

(b) Find the probability that $X_2 = s_3$ given that $X_0 = s_1$.

(c) Given that $X_0 = s_1$, compute the distribution vector $\boldsymbol{\pi}_2$, representing the state probabilities at time step 2.

7. (From 1A—3 points) What is the maximum number of non-zero entries in the transition matrix P of a higher-order Markov chain with states $\mathcal{S} = \{s_1, s_2, \ldots, s_H\}$ and memory m? Your answer should be a function of H and m.

8. (From 1A—4 points) Below is a matrix representing the autocovariance matrix of a random process with two components: $\mathcal{Y}^1$ and $\mathcal{Y}^2$. Information about the autocorrelation functions and cross-correlation functions between the processes are shown for lag values less than 3 in the following table:

Autocorrelation: $R_{\mathcal{Y}^1}(h)$

Lag	Value
0	1.0
1	0.5
2	0.2
3	0.1

Cross-correlation: $R_{\mathcal{Y}^1\mathcal{Y}^2}(h)$

Lag	Value
0	0.8
1	0.4
2	0.2
3	0.1

Cross-correlation: $R_{\mathcal{Y}^2\mathcal{Y}^1}(h)$

Lag	Value
0	0.8
1	0.3
2	0.2
3	0.05

Autocorrelation: $R_{\mathcal{Y}^2}(h)$

Lag	Value
0	1.0
1	0.4
2	0.1
3	0.05

Assuming that the variance of $\mathcal{Y}^1$ and $\mathcal{Y}^2$ are 1 and 2, respectively, answer the following questions:

(a) Calculate the covariance between $\mathcal{Y}^1$ and $\mathcal{Y}^2$ at lag 0.

(b) Compute $R_{\mathcal{Y}^1}(-2)$

(c) Calculate $R_{\mathcal{Y}^1\mathcal{Y}^2}(-1)$?

(d) How much of the total variance of $\mathcal{Y}^1$ is explained by lagged values up to lag 3?