MATHEMATICAL PROOFS AND CONCEPTS IN TIME SERIES ANALYSIS FOR STOCK PREDICTIONS USING ARIMA-LSTM HYBRID MODEL

SHEAN DE FONSEKA

ABSTRACT

This report is inspired by and seeks to derive a detailed understanding from the academic paper titled "Minimum Message Length in Hybrid ARMA and LSTM Model Forecasting" authored by Zheng Fang, David L. Dowe, Shelton Peiris, and Dedi Rosadi. The concepts and mathematical proofs presented in this report aim to show my understanding on the theory, particularly in applying it in the context of financial stock predictions.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Background & Objective

This report looks into the concepts and mathematical proofs in time series analysis, with a particular focus on financial stock predictions. Inspired by the academic paper "Minimum Message Length in Hybrid ARMA and LSTM Model Forecasting" by Zheng Fang, David L. Dowe, Shelton Peiris, and Dedi Rosadi, this paper aims to present my understanding on the theory and mathematical concepts. We explore ARIMA and SARIMA models, emphasising their practical applications in predicting financial stock prices. Additionally, the report investigates the integration of hybrid models, specifically combining ARMA with LSTM, to capture both linear and non-linear patterns in time series data. By deriving mathematical proofs and presenting detailed examples, this report provides a detailed outlook in ARIMA-LSTM hybrid modelling.

2. HOMOGENOUS NON-STATIONARY TIME SERIES

2.1 Definition

Homogeneous Non-Stationarity: Changing mean or variance over time, but the pattern of change is consistent.
For a homogeneous non-stationary time, series, the process that causes these changes in the mean or variance is consistent across the entire series. This can be expressed as:

Equation 2.1.1: $\mu_t = f(t)$
Equation 2.1.2: $\sigma^2{}_t = g(t)$

where f(t) and g(t) are functions that describe how the mean and variance evolve over time, respectively. The key characteristic of homogeneous non-stationarity is that the functional forms of f(t) and g(t) do not change over time.

More specifically:

- Nonstationary is where the time series data does not have a constant mean or variance over time
- Homogenous states that the deviation from its mean or how its variance changes is the same at all points in time.

2.2 Random Walk

In a random walk (which is a classic case of homogeneous non-stationarity):

Equation 2.1.3: $Y_t = Y_{t-1} + \varepsilon_t$

where

Equation 2.1.4: $\varepsilon_t \sim WN(0, \sigma^2)$

The nature of the shocks themselves is consistent. Each shock, ε_t, comes from the same distribution with the same mean and variance. In other words, the consistency is due to the error terms (or shocks) that have a consistent distribution, typically with a mean of zero and some constant variance.

The variance increases as time progresses because the random shocks (ε_t) are accumulating. Even though the overall variance of the series Y_t grows over time (making it non-stationary), the process that generates each new value is stable and doesn't change (it's homogeneous).

3. TIME SERIES DIFFERENCING & THE BACKSHIFT OPERATOR

Differencing is to convert non-stationary time series to stationary.

We denote Y_t as the original non-stationary time series data. This data may exhibit trends or seasonality indicating that its mean and variance change over time. Since ARIMA models require stationary data, Y_t cannot directly be used, and thus, require a transformation.

3.1 First Order Differencing

Equation 3.1.1: $\Delta Y_t = Y_t - Y_{t-1}$

We know that

Equation 3.1.2: $BY_t = Y_{t-1}$

Thus,

Equation 3.1.3: $\Delta Y_t = Y_t - BY_t$

Take out Y_t

Equation 3.1.4: $\Delta Y_t = Y_t(1 - B)$

3.2 Second Order Differencing

Second-order differencing means applying the differencing operation again to the already differenced series ΔY_t.

Equation 3.2.1: $\Delta^2 Y_t = \Delta Y_t(1 - B)$

We substitute

Equation 3.2.2: $\Delta Y_t = Y_t(1 - B)$

to get

Equation 3.2.3: $\Delta^2 Y_t = Y_t(1 - B)(1 - B) = Y_t(1 - B)^2$

Expand the brackets

Equation 3.2.4: $\Delta^2 Y_t = Y_t(1 - 2B + B^2)$

Expand the brackets

Equation 3.2.5: $\Delta^2 Y_t = Y_t - 2BY_t + B^2Y_t$

Remember that BY_t is simply the one value before it in the time series.

Equation 3.2.6: $BY_t = Y_{t-1}$

Therefore, we can derive

Equation 3.2.7: $\Delta^2 Y_t = Y_t - 2Y_{t-1} + Y_{t-2}$

3.3 n^{th} Order Differencing

Using Equations 3.1.4 and Equation 3.2.1, we can derive the formula:

Equation 3.3.1: $\Delta^d Y_t = Y_t(1 - B)^d$

We assign X_t to

Equation 3.3.2: $X_t = Y_t(1 - B)^d$

We denote the stationary time series data as X_t which is used for ARMA modelling. Thus,

Equation 3.3.3: $X_t = Y_t(1 - B)^d$

It indicates that the time series X_t is obtained by differencing the original series Y_t d times.

3.4 Seasonal Order Differencing

Remember Equation 3.1.2, thus, we can derive:

Equation 3.4.1: $B^m Y_t = Y_{t-m}$

The first seasonal difference is

Equation 3.4.2: $(1 - B^m)Y_t = Y_t - Y_{t-m}$

To apply the first difference d times, we raise $1-B^m$ to the power of d. The application of this operator d times effectively iterates the differencing process:

Equation 3.4.3: $(1 - B^m)^d Y_t$

Let's illustrate this with d=2 for simplicity:

Equation 3.4.4: $(1 - B^m)^2 Y_t = (1 - B^m)(1 - B^m) Y_t$

First apply $1-B^m$:

Equation 3.4.5: $(1 - B^m)Y_t = Y_t - Y_{t-m}$

Then apply $1-B^m$ again to the result:

Equation 3.4.6: $(1 - B^m)(Y_t - Y_{t-m}) = Y_t - Y_{t-m} - B^m Y_t + B^m Y_{t-m}$

Equation 3.4.7: $Y_t - Y_{t-m} - B^m Y_t + B^m Y_{t-m} = Y_t - Y_{t-m} - Y_{t-m} + Y_{t-2m} = Y_t - 2Y_{t-m} + Y_{t-2m}$

The second order differencing takes the current value, subtracts twice the value one season back, and adds back the value two seasons back. This ensures that the twice-applied seasonal differences are correctly accounted for, effectively removing any linear trends within the seasonal periods.

4. WHITE NOISE AND ERROR TERMS

4.1 Definition & Mathematical Representation

Equation 4.1.1: $\epsilon_t \sim WN(0, \sigma^2)$

- ε_t: This is a time series, where t indexes time periods. Each ε_t is a random variable representing the value of the series at time t.
- $\sim$: This symbol means "is distributed as."
- WN: This stands for "White Noise," which is a statistical process that has a mean of zero and is not autocorrelated; in other words, there is no discernible pattern or predictable trend in the series.
- $(0, \sigma^2)$: These are the parameters of the white noise process.

The first parameter, 0, is the mean of the white noise process.
The second parameter, σ^2, is the variance of the white noise process. It implies that every term in the series has the same variance and that variance does not change over time.

Concept in time series analysis because it represents the ideal form of randomness with no serial dependence.

It can follow any distribution as long as it meets the criteria of having a mean of zero, constant variance, and no autocorrelation among variables ε_t at different time lags.
The distribution could be normal, but it could also be uniform, binomial, Poisson, or any other distribution, depending on the context and the specifics of the process being modelled.

5. AUTOREGRESSIVE (AR) & MOVING AVERAGE (MA) MODELS

5.1 Autoregressive (AR)

AR (auto regressive) means the model is attempting to predict future values based on past values.
It requires the time series data to be stationary.

The AR part of the model uses actual past values of the time series to predict the future. It assumes that there's a direct relationship between past values and future values.

An AR model is defined as (this is for p=1):

Equation 5.1.1: $X_t = c + \phi_1 X_{t-1} + \epsilon_t$

- X_t is the value at time step t
- c is the constant
- ϕ is the coefficient
- ε_t is a white noise error term with $N(0, \sigma^2)$

The AR model can take parameter p which tells us how many prior time steps to use when predicting the current time step.

AR(p) can be expressed as:

Equation 5.1.2: $X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$

where ϕ is the respective coefficient for each prior time step X_{t-i}

5.2 Deriving ϕ(B) (AR Compact Form)

The expression $\phi(B)X_t$ is shorthand for applying the AR(p) model to the time series X_t.

Let's say we have an AR(2) model for clarity. The model would be:

Equation 5.2.1: $X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \epsilon_t$

We could rewrite this using the backshift operator as:

Equation 5.2.2: $X_t = c + \phi_1 B X_t + \phi_2 B^2 X_t + \epsilon_t$

We move X_t over to the LHS

Equation 5.2.3: $X_t - \phi_1 B X_t - \phi_2 B^2 X_t = \epsilon_t + c$

We take out X_t as a common factor

Equation 5.2.4: $X_t(1 - \phi_1 B - \phi_2 B^2) = \epsilon_t + c$

And now we assign ϕ (B) the following expression (for simplicity's sake – assuming backshift=2):

Equation 5.2.5: $\phi(B) = (1 - \phi_1 B - \phi_2 B^2)$

Equation 5.2.6: $\phi(B) = \left(1 - \phi_1 B - \cdots - \phi_p B^p\right) = 1 - \sum_{j=1}^{p} \phi_j B^j$

Equation 5.2.7: $\phi(B)X_t = X_t(1 - \phi_1 B - \phi_2 B^2) = X_t + X_t B\phi_1 + X_t B^2 \phi_2$

Thus, the equation can be rewritten as:

Equation 5.2.8: $\phi(B)X_t = \epsilon_t + c$

The left-hand side, $\phi(B)X_t$, represents the part of the current value of the time series that is explained by its own past values.

Now to derive ϵ_t, we look at the derivation of MA below.

5.3 Moving Average (MA)

MA (moving average) means the model attempts to predict the future errors based on past errors.
Note: This is not the moving average (smoothing process)

The MA (Moving Average) part uses past forecast errors.
These are the differences between the actual values and what was predicted by the model at previous time steps.
The model assumes these errors contain information that can help in making future predictions.

An MA model is defined as (q=1):

Equation 5.3.1: $X_t = \epsilon_t + \theta_1 \epsilon_{t-1}$

remember this tries to predict the error from the past forecasts (the past errors). Please note that ϵ_t represents the current white noise (the current shock that cannot be predicted). In other words, the model is saying that it is influenced by past errors (MA part), plus a new shock ϵ_t.

Equation 5.3.2: $X_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_p\epsilon_{t-q}$

ϵ_t is the value at time step t, c is a constant, θ_1 is a coefficient and epsilon t-1 is a previous white noise error term

Equation 5.3.3: $X_t = \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$

θ_j is the corresponding coefficient for each respective prior error t-j

5.4 Deriving θ(B) (MA Compact Form)

The expression $\theta(B)X_t$ is shorthand for applying the MA(q) model to the time series ε_t.

Let's say we have an MA(2) model for clarity. The model would be:

Equation 5.4.1: $X_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2}$

We could rewrite this using the backshift operator as:

Equation 5.4.2: $X_t = \epsilon_t + \theta_1 B\epsilon_t + \theta_2 B^2\epsilon_t$

We take out ε_t as a common factor

Equation 5.4.3: $\epsilon_t(1 + \theta_1 B + \theta_2 B^2) = X_t$

Equation 5.4.4: $(1 + \theta_1 B + \theta_2 B^2)\epsilon_t = X_t$

And now we assign θ(B) the following expression (for simplicity's sake):

Equation 5.4.5: $\theta(B) = (1 + \theta_1 B + \theta_2 B^2)$

Equation 5.4.5: $\theta(B)\varepsilon_t = \varepsilon_t(1 + \theta_1 B + \theta_2 B^2) = \epsilon_t + \varepsilon_t B\theta_1 + \varepsilon_t B^2\theta_2$

Thus, the equation can be rewritten as:

Equation 5.4.6: $\theta(B)\epsilon_t = (1 + \theta_1 B + \theta_2 B^2)\epsilon_t = X_t$

We have derived:

Equation 5.4.7: $X_t = \theta(B)\epsilon_t$

This means that the value at time t, X_t, is the sum of the current and past white noise (error) terms, which are weighted by the coefficients θ_1 and θ_2.

6. ARMA MODELS

6.1 Combining AR and MA

It is simply merging the two AR and MA models together.

Note: It only works on stationary time series (X_t).
The means the mean and variance do not change over time.
Given that most of the real-world problems are not stationary, we have to transform the data to stationary data.
This is known as integration.

It leverages both the actual past values and the information contained in the past forecast errors to make predictions.
This is why ARMA models can be more powerful than using AR or MA alone.

Equation 6.1.1: $X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_i \epsilon_{t-j}$

p and q are the orders of the AR and MA models respectively.
Remember, ϵ_t represents the current white noise (the current shock that cannot be predicted). In other words, the model is saying that X_t is influenced by its past values (AR part) and past errors (MA part), plus a new shock ϵ_t.

Why does ARMA require stationary time series data?

- Stationarity implies that the statistical properties of the time series (mean, variance, autocorrelation, etc.) are constant over time.
- This assumption is crucial for the ARMA model because it relies on the past values to predict future values. If these properties change over time, the model's coefficients, which are estimated from historical data, it may not be applicable or accurate for future periods.

6.2 ARMA Compact Form Mathematical Proof

The full ARMA(p,q) model is given by:

Equation 6.2.1: $X_t = c + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q}$

This can be rewritten in terms of the backshift operator B as:

Equation 6.2.2: $X_t = c + \phi_1 B X_t + \cdots + \phi_p B^p X_t + \epsilon_t + \theta_1 B \epsilon_t + \cdots + \theta_q B^q \epsilon_t + \epsilon_t$

Group in brackets X_t and ε_t

Equation 6.2.3: $X_t = c + \left(X_t \phi_1 B + \cdots + \phi_p B^p X_t\right) + \left(\epsilon_t + \theta_1 B \epsilon_t + \cdots + \theta_q B^q \epsilon_t\right)$

Move the $X_t\phi$ expression on the RHS to the LHS with X_t

Equation 6.2.4: $X_t - \left(X_t \phi_1 B + \cdots + \phi_p B^p X_t\right) = c + \left(\epsilon_t + \theta_1 B \epsilon_t + \cdots + \theta_q B^q \epsilon_t\right)$
Equation 6.2.5: $X_t - \phi_1 B X_t - \cdots - \phi_p B^p X_t = c + \left(\epsilon_t + \theta_1 B \epsilon_t + \cdots + \theta_q B^q \epsilon_t\right)$

The LHS now has the complete $\phi(B)X_t$ (Equation 5.2.8), so we just substitute this

Equation 6.2.6: $\phi(B) X_t = c + \left(\epsilon_t + \theta_1 B \epsilon_t + \cdots + \theta_q B^q \epsilon_t\right)$

Now we replace the ε_t expression on the RHS to $\theta(B)\varepsilon_t$ (Equation 5.4.5)

Equation 6.2.7: $\phi(B) X_t = c + \theta(B) \epsilon_t$

To finalise this, we substitute Equation 3.3.3 into this

Equation 6.2.8: $\phi(B)(1 - B)^d Y_t = c + \theta(B) \epsilon_t$

7. ARIMA MODELS

7.1 Introduction

ARIMA adds the differencing components to the ARMA. This allows ARIMA to model non-stationary time series. Remember, ARMA is only allowed on stationary time series.

The differencing step essentially subtracts the previous observation from the current observation, potentially multiple times (d times), to make the series stationary.

Once stationary, the AR and MA components can model the data.

$$\text{ARIMA}(p, d, q)$$

- p (AR order)
- d (degree of differencing)
- q (MA order)

The integration part (differencing) makes ARIMA capable of handling data with trends and removing seasonality, thereby broadening its applicability to a wider range of time series data that ARMA cannot handle directly.

8. SEASONAL ARIMA (SARIMA) MODELS

8.1 Introduction

SARIMA extends the ARIMA model by incorporating an additional seasonal component. SARIMA models account for both regular (non-seasonal) patterns and seasonal patterns.

This seasonal component mirrors the structure of the non-seasonal ARMA part (AR and MA), but it operates on a seasonal timescale.

$$Y_t \sim \text{SARIMA}(p,d,q)(P,D,Q)_m$$

$(p, d, q)_m$ is for the non-seasonal component.
$(P, D, Q)_m$ is for the seasonal component.

m = number of observations per year
P = Number of seasonal AR terms;
D = Number of seasonal differences
Q = Number of seasonal MA terms

Interpretation
In simple terms, SAR is the same as AR except that it's lags are dependent on the last seasonal trend (p = m) instead of just p lags. Similarly, MAR is the same as MAR except that it's lags for calculating errors is dependent on the last seasonal trend (q = m) instead of just q lags.

8.2 Deriving Φ(B) (SAR Compact Form)

This is very similar to Section 5.2 when deriving $\phi(B)X_t$.

Let's say we have an SAR(2) model for clarity. The model would be:

Equation 8.2.1: $X_t = c + \Phi_1 X_{t-m} + \Phi_2 X_{t-2m} + \epsilon_t$

We could rewrite this using the backshift operator as:

Equation 8.2.2: $X_t = c + \Phi_1 B^m X_t + \Phi_2 B^{2m} X_t + \epsilon_t$

We move X_t over to the LHS

Equation 8.2.3: $X_t - \Phi_1 B^m X_t - \Phi_2 B^{2m} X_t = \epsilon_t + c$

We take out X_t as a common factor

Equation 8.2.4: $X_t(1 - \Phi_1 B^m - \Phi_2 B^{2m}) = \epsilon_t + c$

And now we assign Φ(B) the following expression (for simplicity's sake – assuming backshift=2):

Equation 8.2.5: $\Phi(B^m) = (1 - \Phi_1 B^m - \cdots - \Phi_P B^{Pm})$
Equation 8.2.6: $\Phi(B)X_t = X_t(1 - \Phi_1 B^m - \cdots - \phi_P B^{Pm}) = X_t - X_t B^m \Phi_1 - \cdots - X_t B^{Pm} \Phi_P$

Thus, the equation can be rewritten as:

Equation 8.2.7: $X_t \Phi(B^m) = \epsilon_t + c$

The left-hand side, $\Phi(B)X_t$, represents the part of the current value of the time series that is explained by its own past values.

8.3 Deriving Θ(B) (SAR Compact Form)

This is very similar to Section 5.4 when deriving $\theta(B)X_t$.

Let's say we have an SAR(2) model for clarity. The model would be:

Equation 8.3.1: $X_t = \epsilon_t + \Theta_1 \epsilon_{t-m} + \Theta_2 \epsilon_{t-2m}$

We could rewrite this using the backshift operator as:

Equation 8.3.2: $X_t = \epsilon_t + \Theta_1 B^m \epsilon_t + \Theta_2 B^{2m} \epsilon_t$

We take out ϵ_t as a common factor

Equation 8.3.3: $\epsilon_t(1 + \Theta_1 B^m + \Theta_2 B^{2m}) = X_t$

And now we assign $\Theta(B^m)$ the following expression (for simplicity's sake):

Equation 8.3.4: $\Theta(B^m) = \left(1 + \Theta_1 B^m + \cdots + \Theta_Q B^{Qm}\right)$
Equation 8.3.5: $\Theta(B^m)\epsilon_t = \epsilon_t\left(1 - \Theta_1 B^m - \cdots - \Theta_Q B^{Qm}\right) = \epsilon_t + \epsilon_t B^m \Theta_1 + \epsilon_t B^{2m} \Theta_2$

Thus, the equation can be rewritten as:

Equation 8.3.6: $\epsilon_t \Theta(B^m) = X_t$

The left-hand side, $\Theta(B^m)\ \epsilon_t$, represents the part of the current value of the time series that is explained by its own past values.

8.4 Seasonal Component of SARIMA Model

The full seasonal lagged ARMA(p,q) model is given by:

Equation 8.4.1: $X_t = c + \Phi_1 X_{t-m} + \cdots + \Phi_P X_{t-Pm} + \epsilon_t + \Theta_1 \epsilon_{t-m} + \cdots + \Theta_Q \epsilon_{t-Qm}$

This can be rewritten in terms of the backshift operator B as:

Equation 8.4.2: $X_t = c + \Phi_1 B^m X_t + \cdots + \Phi_P B^{Pm} X_t + \epsilon_t + \Theta_1 B^m \epsilon_t + \cdots + \Theta_Q B^{Qm} \epsilon_t + \epsilon_t$

Group in brackets X_t and ε_t

Equation 8.4.3: $X_t = c + (X_t \Phi_1 B^m + \cdots + \Phi_P B^{Pm} X_t) + \left(\epsilon_t + \Theta_1 B^m \epsilon_t + \cdots + \Theta_Q B^{Qm} \epsilon_t\right)$

Move the $X_t\phi$ expression on the RHS to the LHS with X_t

Equation 8.4.4: $X_t - (X_t \Phi_1 B^m + \cdots + \Phi_P B^{Pm} X_t) = c + \left(\epsilon_t + \Theta_1 B^m \epsilon_t + \cdots + \Theta_Q B^{Qm} \epsilon_t\right)$
Equation 8.4.5: $X_t - \Phi_1 B^m X_t - \cdots - \Phi_P B^{Pm} X_t = c + \left(\epsilon_t + \Theta_1 B^m \epsilon_t + \cdots + \Theta_Q B^{Qm} \epsilon_t\right)$

The LHS now has the complete $\Phi(B^m)X_t$ (Equation 8.2.6), so we just substitute this

Equation 8.4.6: $\Phi(B^m)X_t = c + \left(\epsilon_t + \Theta_1 B^m \epsilon_t + \cdots + \Theta_q B^q \epsilon_t\right)$

Now we replace the ε_t expression on the RHS to $\Theta(B^m)\varepsilon_t$ (Equation 8.3.5)

Equation 8.4.7: $\Phi(B^m)X_t = c + \Theta(B^m)\epsilon_t$

To finalise this, we substitute Equation 3.4.3 into this

Equation 8.4.8: $\Phi(B^m)(1 - B^m)^d Y_t = c + \Theta(B^m)\epsilon_t$

8.5 SARIMA Compact Form Mathematical Proof

Remember we have Equation 8.4.7 and Equation 6.2.8.
Firstly, we must adjust the second equation. Traditional time series models like ARIMA assume that the error components (ϵ_t, ϵ_{t-1}, ...) are uncorrelated white noise (WN). However, this assumption often does not hold, especially in complex time series data such as perhaps with stock predictions, where errors from similar periods (e.g., the same month across different years) may show correlation. Hence, we redefine the error terms in the equations.

We redefine ϵ_t as an ARMA model (forecasting error terms) for the SARIMA model.
We redefine the white noise process for errors as α_t.

Therefore, we replace Y_t with ϵ_t and ϵ_t (which was originally WN) as α_t.
This states that the error dynamics themselves are modelled as an ARMA process, where the differenced errors (ϵ_t) are expressed through an autoregressive moving average structure dictated by $\phi(B)$ and $\theta(B)$.

By modelling the errors ϵ_t with their own ARMA process, the approach aims to capture more complex structures and dependencies that are not accounted for in the primary model. This is represented by the formula below.

Equation 8.5.1: $\phi(B)(1-B)^d\epsilon_t = c + \theta(B)\alpha_t$

For the sake of simplicity and derivation, we will ignore the constants

Equation 8.5.2: $\Phi(B^m)(1-B^m)^dY_t = \Theta(B^m)\epsilon_t$
Equation 8.5.3: $\phi(B)(1-B)^d\epsilon_t = \theta(B)\alpha_t$

We move $\phi(B)(1-B)^d$ to the RHS to isolate ϵ_t.

Equation 8.5.4: $\epsilon_t = \theta(B)\alpha_t[\phi(B)(1-B)^d]^{-1}$

We substitute Equation 8.5.4 into Equation 8.5.2 to get

Equation 8.5.5: $\Phi(B^m)(1-B^m)^dY_t = \Theta(B^m)[\theta(B)\alpha_t[\phi(B)(1-B)^d]^{-1}]$

We expand the negative exponent

Equation 8.5.6: $\Phi(B^m)(1-B^m)^dY_t = \Theta(B^m)\left[\theta(B)\alpha_t\phi(\mathrm{B})^{-1}(1-\mathrm{B})^{-\mathrm{d}}\right]$

We expand the square brackets

Equation 8.5.7: $\Phi(B^m)(1-B^m)^dY_t = \Theta(B^m)\theta(B)\alpha_t\phi(\mathrm{B})^{-1}(1-\mathrm{B})^{-\mathrm{d}}$

We move the negative exponents to the LHS

Equation 8.5.8: $\phi(\mathrm{B})\Phi(B^m)(1-B^m)^d(1-\mathrm{B})^{\mathrm{d}}Y_t = \Theta(B^m)\theta(B)\alpha_t$

9. THEORETICAL FOUNDATION OF STATISTICAL MODELLING

9.1 Probability Models and Likelihood Functions

Assume we have a dataset $X = \{x_1, x_2, \ldots, x_n\}$ where each observation is independently and identically distributed. We have a probability model (from PDF) with parameters θ which we use for the likelihood function. The larger the probability, the more likely the observed data would be to arise under that model.

Given this probability density function $f_\theta(y_i|\theta)$, we can write the likelihood function as follows:

Equation 9.1.1: $p(y|\theta) = \prod_{i=1}^{n} f_\theta(y_i|\theta)$

It is important to remember the following properties of the PDF (probability density function) which we define as the probability model, has the following properties:

9.1.1 Rule 1: Total Probability Over Sample Space X is 1

(1) Total probability over the entire sample space X is 1:

Equation 9.1.1.1: $\int_{x\in X} f_\theta(y)dx = 1$

9.1.2 Rule 2: Leibniz Integral Rule

Also called differentiation under the integral sign), we can interchange the derivative and the integral as long as $f_\theta(x)$ is sufficiently smooth (i.e., continuous, and differentiable with respect to both x and θ):

Equation 9.1.2.1: $\frac{\partial}{\partial\theta}\int_{x\in X} f_\theta(y)dx = \int_{x\in X}\frac{\partial}{\partial\theta} f_\theta(y)dx$

Therefore, we can write based on the rule:

$$\text{Equation 9.1.2.2: } \frac{\partial}{\partial\theta}\int_{x\in X} f_\theta(y)dx = \int_{x\in X}\frac{\partial}{\partial\theta} f_\theta(y)dx = 0$$

$$\text{Equation 9.1.2.3: } \frac{\partial^2}{\partial\theta^2}\int_{x\in X} f_\theta(y)dx = \int_{x\in X}\frac{\partial^2}{\partial\theta^2} f_\theta(y)dx = 0$$

The total probability (1) does not change with respect to θ, so the derivative of a constant (1) is zero.

9.2 Maximum Likelihood Estimation (MLE)

9.2.1 Maximum Likelihood

The method of maximum likelihood says we should use the model that assigns the greatest probability to the data we have observed. Formally, the maximum likelihood (ML) estimator is found by solving:

$$\text{Equation 9.2.1.1: } \hat{\theta} = \arg\max_{\theta}\{p(y|\theta)\}$$

Where $p(y|\theta)$ is the likelihood function

9.2.2 Negative Log-Likelihood

We define the following:

$$\text{Equation 9.2.2.1: } \mathcal{L}(y|\theta) = -\log\left[p(y|\theta)\right]$$

It is mathematically easier to solve the equivalent problem:

$$\text{Equation 9.2.2.2: } \hat{\theta} = \arg\min_{\theta}\{-\log\left[p(y|\theta)\right]\}$$

Where $p(y|\theta)$ is the likelihood function

9.3 The Score Function and Its Properties

The score function, denoted $S(\theta \mid x)$, is the first derivative of the log-likelihood function with respect to θ:

$$\text{Equation 9.3.1: } S(\,\theta \mid y\,) = \frac{d}{d\theta}(\log[p(y|\theta)])$$

Note: We don't apply the negative log likelihood here, only logarithm.

Intuitively, if you think about the likelihood function as a measure of how "plausible" different values of θ are given the observed data X, the score tells you which direction to adjust θ to increase that plausibility.

At the true value of θ_0, the likelihood function is maximally aligned with the true data-generating process. This alignment suggests that there should be no systematic tendency to increase or decrease the likelihood by nudging θ slightly in either direction. Therefore, the average rate of change (or slope) of the likelihood at θ0 is zero, which leads to the expectation of the score being zero.

The score function is a vector-valued function when θ is a vector (m-dimensional), with each element corresponding to the partial derivative of the log-likelihood with respect to a parameter:

$$\text{Equation 9.3.2: } S(\theta) = \begin{bmatrix} \frac{d}{d\theta_1}(\log[p(y|\theta)]) \\ \vdots \\ \frac{d}{d\theta_m}(\log[p(y|\theta)]) \end{bmatrix}$$

This differentiation yields a $(1 \times m)$ row vector at each value of θ and x and indicates the sensitivity of the likelihood (its derivative normalised by its value). The score function represents how sensitive the likelihood function is to changes in θ.

At the true parameter value (θ_0), the expected value of the score function is zero:

$$\text{Equation 9.3.3: } \mathbb{E}_{\theta_0}[S(\theta)] = 0$$

Fisher's score and maximum likelihood estimation is to find a parameter value that would set the gradient equal to zero. This is exactly what I had thought, but there are subtle intricacies taking place here that deserves our attention.

These zero expectations reflect that, on average, there is no systematic direction to adjust θ since the likelihood function is maximised at the true value.

In other assumptions, the following can also be assumed more accurately (for the proof):

$$\text{Equation 9.3.4: } \mathbb{E}_{\theta}[S(\theta)] = \mathbb{E}_{\theta}\left[\frac{d}{d\theta}(\log[p(y|\theta)])\right] = 0$$

This follows from the fact that the integral of the derivative of a probability density function over its support is zero, provided we can interchange differentiation and integration (Regularity Condition 2) as per the (2) Leibniz Integral Rule as defined above.

9.4 Variance-Covariance Matrix

Variance is to consider it as a measure of how far samples are from the mean.
We square that quantity to prevent negative values from cancelling out positive ones.

$$\text{Equation 9.4.1: } Var(X) = \mathbb{E}[(X-\mu)^2] = \mathbb{E}[(X-E[X])^2] = \mathbb{E}[X^2] - E[X]^2$$

Covariance is just an extension of the concept applied to compare two random variables instead of one.
We consider how two variables move in tandem with each other.

$$\text{Equation 9.4.2: } Cov[X,Y] = \mathbb{E}\big[(X-\mu_x)\big(Y-\mu_y\big)\big] = \mathbb{E}[(X-E[X])(Y-E[Y])] = E[XY] - E[X]E[Y]$$

The variance-covariance matrix is simply a matrix that contains information on the covariance of multiple random variables in a neat, compact matrix form.

$$\text{Equation 9.4.3: } K = \begin{pmatrix} Cov[X_1,X_1] & Cov[X_1,X_2]\cdots & Cov[X_1,X_n] \\ Cov[X_2,X_1] & Cov[X_2,X_2]\cdots & Cov[X_2,X_n] \\ \vdots & \vdots \ddots & \vdots \\ Cov[X_n,X_1] & Cov[X_n,X_2]\cdots & Cov[X_n,X_n] \end{pmatrix}$$

This can be written in a closed form expression given random vector X where $X = (X_1, X_2, \ldots, X_n)$.

$$\text{Equation 9.4.4: } K = \mathbb{E}[(X - \mathbb{E}[X])(X - E[X]^T)]$$

9.5 Hessian Matrix

The Hessian will also be easy to understand. It is the derivative of the gradient of a scalar-valued function $f: \mathbb{R}^n \to \mathbb{R}$.
For instance :

$$\text{Equation 9.5.1: } \boldsymbol{H}\, f(x,y) = \nabla^2 f(x,y) = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$$

9.6 Jacobian Matrix

The Jacobian is a matrix that holds all first-order partial derivatives of a vector-valued function ($f: \mathbb{R}^n \to \mathbb{R}$):

he Jacobian form of a vector-valued function $h(f(x,y), g(x,y))$ is therefore the following:

$$\text{Equation 9.6.1: } \boldsymbol{J}\, h(f(x,y), g(x,y)) = \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix}$$

9.7 Fisher Information

9.7.1 Fisher Information

Fisher Information measures the amount of information that an observable data set carries about unknown parameters of a model that were used to generate the data. Essentially, it quantifies how well we can expect to estimate these parameters based on the observed data.

Fisher Information comes from how sensitive this log-likelihood function is to changes in the parameter θ. Specifically, it looks at the derivative of the log-likelihood function with respect to θ.

The idea is if a small change in θ leads to a large change in the log-likelihood, the data x is very informative about θ.

$$\text{Equation 9.7.1.1: } F(\theta) = -\mathbb{E}\left[H_{\log[p(x|\theta)]}\right] = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^2}\log[p(y|\theta)]\right]$$

9.7.2 Fisher Information Relationship with Score Function

Under the regularity conditions, the Fisher Information can be expressed under the assumption of the regulatory rules which are:

- The partial derivative of $p(y|\theta)$ with respect to θ exists almost everywhere for almost all y. Even if the derivative fails to exist on a null set (a set of zero probability), the expectation can still be defined.
- The integral of $p(y|\theta)$ can be differentiated under the integral sign with respect to θ (the Leibniz rule mentioned in the report).

$$\text{Equation 9.7.2.1: } F(\theta) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta^2}\log p(y|\theta)\right]$$

$$\text{Equation 9.7.2.2: } F(\theta) = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log p(y|\theta)\right)^2\right]$$

We can adjust this replace X with $S(\theta)$. Remember, also we proved above that the $\mathbb{E}_{\theta_0}[S(\theta)] = 0$, therefore, we have:

$$\text{Equation 9.7.2.3: } K = \mathbb{E}[(S(\theta) - 0)(S(\theta) - 0)] = \mathbb{E}\left[\left(S(\theta)\right)(S(\theta)^T)\right]$$

And this can be rewritten as:

$$\text{Equation 9.7.1.4: } K = \begin{pmatrix} Cov[S(\theta_1), S(\theta_1)] & Cov[S(\theta_1), S(\theta_2)] \cdots & Cov[S(\theta_1), S(\theta_n)] \\ Cov[S(\theta_2), S(\theta_1)] & Cov[S(\theta_2), S(\theta_2)] \cdots & Cov[S(\theta_2), S(\theta_n)] \\ \vdots & \vdots \ddots & \vdots \\ Cov[S(\theta_n), S(\theta_1)] & Cov[S(\theta_n), S(\theta_2)] \cdots & Cov[S(\theta_n), S(\theta_n)] \end{pmatrix}$$

In this matrix form:

Each element $[F(\theta)]_{i,j} = Cov\left[S_i(\theta) \cdot S_j(\theta)\right] = \mathbb{E}[S_i(\theta) \cdot S_j(\theta)]$

Hence, for a scalar parameter θ, this reduces to the single value:

$$\text{Equation 9.7.2.5: } F(\theta) = \mathbb{E}[S(\theta)^2]$$

9.7.3 Fisher Information Matrix Proof

$$\text{Equation 9.7.3.1: } \frac{\partial^2}{\partial\theta^2}\log[p(y|\theta)]$$

To do this, we remember derivative of log is:

$$\frac{d}{dx}\log\big(f(x)\big) = \frac{1}{f(x)} \cdot f'(x)$$

Thus,

$$\text{Equation 9.7.3.2: } \frac{\partial}{\partial\theta}\log[p(y|\theta)] = \frac{1}{p(y|\theta)} \cdot \frac{\partial}{\partial\theta}p(y|\theta) = \frac{\frac{\partial}{\partial\theta}p(y|\theta)}{p(y|\theta)}$$

This step requires Regularity Condition 1, ensuring that $\frac{d}{d\theta}(p(y|\theta))$ exists almost everywhere.

Therefore, we apply the quotient rule:

$$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)\cdot f'(x) - f(x)\cdot g'(x)}{\big(g(x)\big)^2}$$

Thus, we do the following:

$$\frac{\partial}{\partial\theta}\left[\frac{\frac{\partial}{\partial\theta}p(y|\theta)}{p(y|\theta)}\right]$$

$$f(x) = \frac{\partial}{\partial\theta}p(y|\theta)$$

$$f'(x) = \frac{\partial^2}{\partial\theta^2}p(y|\theta)$$

$$g(x) = p(y|\theta)$$

$$g'(x) = \frac{\partial}{\partial\theta}p(y|\theta)$$

$$\text{Equation 9.7.3.3: } \frac{\partial}{\partial\theta}\left[\frac{\frac{\partial}{\partial\theta}p(y|\theta)}{p(y|\theta)}\right] = \frac{\left(p(y|\theta)\cdot\frac{\partial^2}{\partial\theta^2}p(y|\theta)\right) - \left(\frac{\partial}{\partial\theta}p(y|\theta)\cdot\frac{\partial}{\partial\theta}p(y|\theta)\right)}{p(y|\theta)^2}$$

$$= \frac{\left(p(y|\theta)\cdot\frac{\partial^2}{\partial\theta^2}p(y|\theta)\right) - \left(\frac{\partial}{\partial\theta}p(y|\theta)\right)^2}{p(y|\theta)^2} = \frac{\frac{\partial^2}{\partial\theta^2}p(y|\theta)}{p(y|\theta)} - \frac{\left(\frac{\partial}{\partial\theta}p(y|\theta)\right)^2}{p(y|\theta)^2}$$

Again, this step relies on Regularity Condition 1 for the existence of the second derivative.

Now we do:

$$\text{Equation 9.7.3.4: } \mathbb{E}\left[\frac{\partial^2}{\partial\theta^2}\log[p(y|\theta)]\right] = \mathbb{E}\left[\frac{\frac{\partial^2}{\partial\theta^2}p(y|\theta)}{p(y|\theta)} - \frac{\left(\frac{\partial}{\partial\theta}p(y|\theta)\right)^2}{p(y|\theta)^2}\right]$$

Using the linearity of expectation rule

$$E[X - Y] = E[X] - E[Y]$$

We get:

$$\text{Equation 9.7.3.5: } \mathbb{E}\left[\frac{\frac{\partial^2}{\partial\theta^2}p(y|\theta)}{p(y|\theta)} - \frac{\left(\frac{\partial}{\partial\theta}p(y|\theta)\right)^2}{p(y|\theta)^2}\right] = \mathbb{E}\left[\frac{\frac{\partial^2}{\partial\theta^2}p(y|\theta)}{p(y|\theta)}\right] - \mathbb{E}\left[\frac{\left(\frac{\partial}{\partial\theta}p(y|\theta)\right)^2}{p(y|\theta)^2}\right]$$

For the first term, given that we assume that this is continuous random variables (RVs), we can use the rule:

$$E[f(X)] = \int f(x_i)\cdot p(x_i)\,dx$$

Where $p(x_i)$ is $f(x_i)$'s the probability density function for the respective x_i value under X.

Therefore,

$$\text{Equation 9.7.3.6: } \mathbb{E}\left[\frac{\frac{\partial^2}{\partial\theta^2}p(y|\theta)}{p(y|\theta)}\right] = \int\frac{\frac{\partial^2}{\partial\theta^2}p(y_i|\theta)}{p(y_i|\theta)}\cdot p(y_i|\theta)\,dy$$

The $p(y_i|\theta)$ component cancels out to:

$$\int \frac{\partial^2}{\partial \theta^2} p(y_i|\theta)\, dy$$

Remember how we defined by the second rule of PDF's on the (2) Leibniz Integral rule, the first and second derivative are = 0. Regularity Condition 2 ensures that differentiation under the integral sign is valid.

$$\frac{\partial^2}{\partial \theta^2} \int_{x \in X} f_\theta(y) dx = \int_{x \in X} \frac{\partial^2}{\partial \theta^2} f_\theta(y) dx = 0$$

Thus,

$$\int \frac{\partial^2}{\partial \theta^2} p(y_i|\theta)\, dy = 0$$

Now we look at the second component.

$$\text{Equation 9.7.3.7: } \mathbb{E}\left[\frac{\left(\frac{\partial}{\partial \theta} p(y|\theta)\right)^2}{p(y|\theta)^2}\right] = \mathbb{E}\left[\frac{\frac{\partial}{\partial \theta} p(y|\theta)}{p(y|\theta)} \cdot \frac{\frac{\partial}{\partial \theta} p(y|\theta)}{p(y|\theta)}\right]$$

Remember:

$$\frac{\partial}{\partial \theta} \log[p(y|\theta)] = \frac{\frac{\partial}{\partial \theta} p(y|\theta)}{p(y|\theta)}$$

Thus, we can rewrite:

$$\mathbb{E}\left[\frac{\left(\frac{\partial}{\partial \theta} p(y|\theta)\right)^2}{p(y|\theta)^2}\right] = \mathbb{E}\left[\left(\frac{\partial}{\partial \theta} \log[p(y|\theta)]\right)^2\right]$$

Which is also equal to:

$$S(\theta \mid y) = \frac{d}{d\theta}(\log[p(y|\theta)])$$

$$\mathbb{E}\left[\frac{\left(\frac{\partial}{\partial \theta} p(y|\theta)\right)^2}{p(y|\theta)^2}\right] = \mathbb{E}[S(\theta \mid y) \cdot S(\theta \mid y)] = \mathbb{E}[S(\theta \mid y)^2]$$

Therefore, we get:

$$\text{Equation 9.7.3.8: } F(\theta) = \mathbb{E}\left[\frac{\partial^2}{\partial \theta^2} \log[p(y|\theta)]\right] = 0 - \mathbb{E}[S(\theta \mid y)^2] = -\mathbb{E}[S(\theta \mid y)^2]$$

The Fisher Information is defined as the variance of the score.

Now we can write $F(\theta)$ is equal to:

$$\text{Equation 9.7.3.9: } \mathbb{E}\left[\frac{\partial^2}{\partial \theta^2} \log[p(y|\theta)]\right] = -\mathbb{E}\left[\left(\frac{\partial}{\partial \theta} \log[p(y|\theta)]\right)^2\right]$$

Now we just multiply both sides by -1.

$$-\mathbb{E}\left[\frac{\partial^2}{\partial \theta^2} \log[p(y|\theta)]\right] = \mathbb{E}\left[\left(\frac{\partial}{\partial \theta} \log[p(y|\theta)]\right)^2\right]$$

$$\text{Equation 9.7.3.10: } F(\theta) = -\mathbb{E}\left[\frac{\partial^2}{\partial \theta^2} \log[p(y|\theta)]\right] = \mathbb{E}\left[\left(\frac{\partial}{\partial \theta} \log[p(y|\theta)]\right)^2\right]$$

9.8 Proof of AIC Criterion

AIC

$$AIC = 2k - 2\ln(\hat{L})$$

where:

- k is the number of parameters in the model.
- $\hat{L}$ is the maximum value of the likelihood function for the model.

Mathematical Proof of AIC

θ_k: This is the true parameter vector that defines the model.
$\hat{\theta}_k$: This is the estimated parameter vector, obtained through some estimation procedure (e.g., maximum likelihood estimation).
θ: Sometimes used to denote a general parameter vector.
$\hat{\theta}$: Sometimes used to denote an estimated parameter vector, similar to $\hat{\theta}_k$.
θ_0: This denotes the true parameter vector. It represents the "true" values of the parameters that generated the data according to the true distribution $P(X)$.

9.8.1 Kullback-Leibler (KL) Divergence

The formula is derived using Kullback-Leibler (KL) Divergence and Taylor Series expansion.

The Kullback-Leibler (KL) divergence is a measure from information theory that *quantifies how one probability distribution diverges from a second, reference probability distribution*. It is used to measure the difference between the true data distribution and an approximating model.

Given two probability distributions P and Q, where P is the true distribution and Q is the approximate distribution, the KL divergence from Q to P is defined as:

$$I(P||Q) = \mathbb{E}\left[\log\left(\frac{P(X)}{Q(X)}\right)\right]$$

We define:

- $P(X)$ is the true distribution by which y is generated.
    - It is a constant with respect to θ_k.
- $Q(X)$ or $f(y|\theta_k)$ is the approximation distribution with θ_k parameters.
- $L(y|\theta_k)$ is the likelihood corresponding to $Q(X)$ or $f(y|\theta_k)$.
    - $f(y|\theta_k)$ is the approximating distribution with true parameter θ_k.

The goal is to minimise $f(y|\theta_k)$ that best approximates $P(X)$ by minimising the KL divergence.

We can apply the log rules for division to get:

$$\text{Equation 9.8.1.1: } I(P||Q) = \mathbb{E}\big[\log\big(P(X)\big) - \log\big(f(y|\theta_k)\big)\big]$$

We multiply by 2 (for the sake of simplicity of deriving AIC) on both sides. However, it is noted that multiplied by scalar factor 2 serves only this only purpose.

$$2I(P||Q) = 2 \cdot \mathbb{E}\big[\log\big(P(X)\big) - \log\big(f(y|\theta_k)\big)\big]$$

$$\text{Equation 9.8.1.2: } 2I(P||Q) = 2\mathbb{E}\big[\log\big(P(X)\big)] - 2\mathbb{E}[\log\big(f(y|\theta_k)\big)\big]$$

We move the negative inside:

$$2I(P||Q) = 2\mathbb{E}\big[\log\big(P(X)\big)] + 2\mathbb{E}[-\log\big(f(y|\theta_k)\big)\big]$$

Remember that the likelihood is:

$$L(y|\theta_k) = -\log\big(f(y|\theta_k)\big)$$

Therefore, can substitute:

$$\text{Equation 9.8.1.3: } 2I(P||Q) = 2\mathbb{E}\left[\log\big(P(X)\big)] + 2\mathbb{E}[L(y|\theta_k)\right]$$

9.8.2 Approximating the Expected KL Divergence and Introducing the Bias Term

Remember, the AIC formula is striving to pick the best model from our approximated model.
In this formula using Kl, it includes the true distribution ($P(X)$) in the formula. This doesn't necessarily help us as we don't exactly know it; hence, we remove it. In other words, it doesn't take into account the parameter estimation θ_k which is primarily used in approximation model.

And re-write it in terms of this given that P(X) is ignored as we focus on the part that depends on θ_k:

$$\text{Equation 9.8.2.1: } 2I(P||Q) = 2\mathbb{E}[L(y|\theta_k)]$$

For any parameter $\hat{\theta}_k$, we estimate $d(\hat{\theta}_k)$ using $-2\log(f(y|\theta_k)$.

$$\text{Equation 9.8.2.2: } d(\theta_k) = 2L(y|\theta_k) = -2\log\big(f(y|\theta_k)\big)$$

$d(\theta_k)$ represents the divergence term we aim to minimise.

We estimate $d(\theta_k)$ using $\hat{\theta}_k$, the estimated parameter vector:

$$d(\hat{\theta}_k) = -2\log\left(f(y|\hat{\theta}_k)\right)$$

In order to get a good approximation, we consider the approximation:

$$\text{Equation 9.8.2.3: } \mathbb{E}\big[d(\hat{\theta}_k)\big] = \mathbb{E}[-2\log\big(f\big(y|\hat{\theta}_k\big)\big]$$

By estimating using $\hat{\theta}_k$ instead of the true parameter θ_0, it is going to introduce errors (bias).

$$\text{Equation 9.8.2.4: } bias = \mathbb{E}\big[d(\hat{\theta}_k)\big] - \mathbb{E}[-2\log(f(y|\theta_0)] = k$$

On top of this, there is also another bias introduced as the difference between the expected log-likelihood at the true parameter θ_0 and the estimated parameter $\hat{\theta}_k$.

$$bias = \mathbb{E}[-2\log(f(y|\theta_0)] - \mathbb{E}[-2\log\big(f\big(y|\hat{\theta}_k\big)\big] = k$$

This results in the equation:

$$\text{Equation 9.8.2.5: } \mathbb{E}\big[d(\hat{\theta}_k)\big] = \mathbb{E}[-2\log\big(f\big(y|\hat{\theta}_k\big)\big] + [\mathbb{E}\big[d(\hat{\theta}_k)\big] - \mathbb{E}\big[-2\log(f(y|\theta_0)\big] + \mathbb{E}[-2\log(f(y|\theta_0)] - \mathbb{E}[-2\log\big(f\big(y|\hat{\theta}_k\big)\big]]$$

The two bias expressions are roughly equal to k. k represents the number of parameters in θ_k.

Therefore, we can rewrite bias as:

$$\text{Equation 9.8.2.6: } \mathbb{E}\big[d(\hat{\theta}_k)\big] = \mathbb{E}\big[d(\hat{\theta}_k)\big] + 2k$$

Remember 2k represents the error/bias.

9.8.3 Estimating the Bias Using Taylor Series Expansion and Fisher Information

Now, how and why does the expressions approximate to be k?

(1) $\mathbb{E}[-2\log(f(y|\theta_0)]$ Taylor Series

For a function $f(x)$ of a single variable, the Taylor series expansion around a point x_0 is given by:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \cdots$$

This gives us:

$$\text{Equation 9.8.3.1: } \mathbb{E}[-2\log(f(y|\theta_0)] \approx \mathbb{E}[-2\log\big(f\big(y|\hat{\theta}_k\big)\big] + \big(\hat{\theta}_k - \theta_0\big)^T \nabla L\big(y|\hat{\theta}_k\big) + \frac{1}{2}\big(\hat{\theta}_k - \theta_0\big)\nabla^2 L\big(y|\hat{\theta}_k\big)\big(\hat{\theta}_k - \theta_0\big)^T$$

Note: We transpose $\big(\hat{\theta}_k - \theta_0\big)$ to get the multiplication to work between matrices, it represents the $\big(\hat{\theta}_k - \theta_0\big)^2$ term.

Since θ_0 is the true parameter, the gradient $\nabla L\big(y\big|\hat{\theta}_k\big) = 0$ as per the MLE process, simplifying the expansion to

$$\mathbb{E}[-2\log(f(y|\theta_0)] \approx \mathbb{E}[-2\log(f(y|\hat{\theta}_k)] + \frac{1}{2}(\hat{\theta}_k - \theta_0)^T \nabla^2 \mathbb{E}[-2\log(f(y|\hat{\theta}_k)](\hat{\theta}_k - \theta_0)^T$$

We move $\mathbb{E}[-2\log(f(y|\theta_0)]$ to the LHS getting:

$$\text{Equation 9.8.3.2: } \mathbb{E}[-2\log(f(y|\theta_0)] - \mathbb{E}[-2\log(f(y|\hat{\theta}_k)] \approx \frac{1}{2}(\hat{\theta}_k - \theta_0)^T \nabla^2 \mathbb{E}[-2\log(f(y|\hat{\theta}_k)](\hat{\theta}_k - \theta_0)^T$$

Remember we have the following equation to try and prove that it is equal to k.

$$\mathbb{E}[-2\log(f(y|\theta_0)] - \mathbb{E}[-2\log(f(y|\hat{\theta}_k)] = k$$

The OBSERVED Fisher Information Matrix $\mathcal{F}(\hat{\theta}_k)$ is defined as:

$$\text{Equation 9.8.3.2: } \mathcal{F}(\hat{\theta}_k) = \nabla^2[-f(y|\hat{\theta}_k)]$$

Thus, the expected difference becomes:

$$\text{Equation 9.8.3.3: } \mathbb{E}[-2\log(f(y|\theta_0)] - \mathbb{E}[-2\log(f(y|\hat{\theta}_k)] \approx \frac{1}{2}(\hat{\theta}_k - \theta_0)^T \mathcal{F}(\hat{\theta}_k)(\hat{\theta}_k - \theta_0)^T$$

Applying the theory of generalised chi distribution using quadratic form (in more detail below), it gives:

$$\text{Equation 9.8.3.4: } \frac{1}{2} \cdot \mathbb{E}[(\hat{\theta}_k - \theta_0)^T \mathcal{F}(\hat{\theta}_k)(\hat{\theta}_k - \theta_0)^T = \frac{1}{2} \cdot k = \frac{k}{2}$$

(2) $\mathbb{E}[d(\hat{\theta}_k)]$ Taylor Series

$$\text{Equation 9.8.3.5: } \mathbb{E}[d(\hat{\theta}_k)] = \mathbb{E}[-2\log(f(y|\hat{\theta}_k)]$$

For a function $f(x)$ of a single variable, the Taylor series expansion around a point x_0 is given by:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \cdots$$

This gives us:

$$\mathbb{E}[d(\hat{\theta}_k)] \approx \mathbb{E}[-2\log(f(y|\theta_0)] + (\theta_0 - \hat{\theta}_k)\nabla\mathbb{E}[-2\log(f(y|\theta_0)] + \frac{1}{2}(\theta_0 - \hat{\theta}_k)\nabla^2\mathbb{E}[-2\log(f(y|\theta_0)](\theta_0 - \hat{\theta}_k)^T$$

Since θ_0 is the true parameter, the gradient $\nabla\mathbb{E}[-2\log(f(y|\theta_0)] = 0$, simplifying the expansion to

$$\mathbb{E}[d(\hat{\theta}_k)] \approx \mathbb{E}[-2\log(f(y|\theta_0)] + \frac{1}{2}(\theta_0 - \hat{\theta}_k)\nabla^2\mathbb{E}[-2\log(f(y|\theta_0)](\theta_0 - \hat{\theta}_k)^T$$

We move $\mathbb{E}[-2\log(f(y|\theta_0)]$to the LHS getting:

$$\mathbb{E}[d(\hat{\theta}_k)] - \mathbb{E}[-2\log(f(y|\theta_0)] \approx \frac{1}{2}(\theta_0 - \hat{\theta}_k)\nabla^2\mathbb{E}[-2\log(f(y|\theta_0)](\theta_0 - \hat{\theta}_k)^T$$

Remember we have the following equation to try and prove that it is equal to k.

$$\mathbb{E}[d(\hat{\theta}_k)] - \mathbb{E}[-2\log(f(y|\theta_0)] = k$$

The EXPECTED Fisher Information Matrix $I(\theta_0)$ is defined as:

$$\text{Equation 9.8.3.5: } I(\theta_0) = E[\nabla^2 L(y|\theta_0)] = \mathbb{E}[\nabla^2[-2\log(f(y|\theta_0)]]$$

Thus, the expected difference becomes:

$$\mathbb{E}[d(\hat{\theta}_k)] - \mathbb{E}[-2\log(f(y|\theta_0)] \approx \frac{1}{2}(\hat{\theta}_k - \theta_0)^T I(\theta_0)(\hat{\theta}_k - \theta_0)^T$$

Applying the theory of generalised chi distribution using quadratic form (in more detail below), it gives:

$$\text{Equation 9.8.3.6: } \frac{1}{2} \cdot \mathbb{E}[(\hat{\theta}_k - \theta_0)^T I(\theta_0)(\hat{\theta}_k - \theta_0)^T = \frac{1}{2} \cdot k = \frac{k}{2}$$

Now, to fully understand the theory and conversions please reference the following.

We are to make a few assumptions in order to make this theory work.
(1) The Maximum Likelihood Estimation of θ_k is approximately normally distributed for a large value of n.
(2) The log-likelihood has to be twice differentiable with respect to θ_0.

This is based on the Central Limit Theorem (CLT)
The Central Limit Theorem (CLT) is a fundamental theorem in probability theory that describes the behavior of the sum of a large number of independent and identically distributed (i.i.d.) random variables.

Let $Y_1, Y_2, \ldots, Y_n$ be i.i.d random variables with:

$$\mathbb{E}[Y_i] = \mu$$
$$Var[Y_i] = \sigma^2$$

For a large sample size n, the distribution of the sum $S = Y_1 + Y_2 + \cdots + Y_n$, can be approximated by a normal distribution with mean $n\mu$ and variance $n\sigma^2$. The approximation gets better and better for increasing n.

Hence, mathematically,

$$\sum_{i=1}^{n} Y_i \xrightarrow{d} N(n\mu, n\sigma^2)$$

Where $\xrightarrow{d}$ denotes convergence in distribution.

In other words, as n→∞, we can conclude that

$$\bar{Y} = \sum_{i=1}^{n} Y_i$$

$$\bar{Y} \xrightarrow{d} N(\mu, n\sigma^2)$$

The MLE θ_k is a random variable because it is calculated from sample data, which are random variables. Each time we draw a new sample from the population, the resulting MLE θ_k will likely differ because it depends on the specific sample drawn. This inherent variability makes θ_k a random variable.

Following the Central Limit Theorem (CLT), for large sample sizes (n→∞), the distribution of the MLE θ_k can be approximated by a normal distribution centered around the true parameter value θ_0. This result is derived from the Central Limit Theorem applied in the context of maximum likelihood estimation.

$$\theta_k \xrightarrow{d} N\left(\theta_0, I^{-1}(\theta_0)\right)$$

Why is variance of θ_k equal to the inverse of Fisher Information Matrix?

It is important to remember that the expectation of the score function (the first derived of log likelihood with respect to the actual parameter) is zero at the true parameter value θ_0.

And that the MLE θ_k, the score function is = 0

$$U(\theta_k) = 0$$

Where $U(\theta_k)$ is the first derivative of the log likelihood function with respect to θ_k.
Remember, this is done in order to get the best parameters for the model.

Thus, we do the derivative here (only first order):

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \cdots$$

$$f(y|\theta_k) = f(y|\theta_0) + f'(\theta_0)(\theta_k - \theta_0)$$

We find the derivative of this, which transforms to

Equation 9.8.3.7: $U(\theta_k) = \frac{\partial f(y|\theta_k)}{\partial \theta_k} \approx \frac{\partial f(y|\theta_0)}{\partial \theta_0} + \frac{\partial \left[\frac{\partial f(y|\theta_0)}{\partial \theta_0}\right]}{\partial \theta_0}(\theta_k - \theta_0) = U(\theta_0) + \frac{\partial [U(\theta_0)]}{\partial \theta_0}(\theta_k - \theta_0)$

And given that $U(\theta_k) = 0$, we get:

$$U(\theta_0) + \frac{\partial [U(\theta_0)]}{\partial \theta_0}(\theta_k - \theta_0) = 0$$

On top of that, remember the 2nd order derivative of the true parameter is Fisher Information Matrix (θ_0), thus, $\frac{\partial [U(\theta_0)]}{\partial \theta_0} = I(\theta_0)$. Thus,

$$U(\theta_0) + I(\theta_0)(\theta_k - \theta_0) = 0$$

We move $U(\theta_0)$ and $I(\theta_0)$ to the other side

$$I(\theta_0)(\theta_k - \theta_0) = -U(\theta_0)$$

$$\theta_k - \theta_0 = -U(\theta_0) \cdot I^{-1}(\theta_0)$$

We take the variance of both sides:

$$Var(\theta_k - \theta_0) = Var\left(-U(\theta_0) \cdot I^{-1}(\theta_0)\right)$$

Since $I(\theta_0)$ is a constant with respect to θ_0, we can factor it out (remember the variance rule when taking out scalar factors). Note, we take the negative sign out with I (which is squared, making it positive).

$$V[cX] = c^2 \cdot V[X]$$

$$Var(\theta_k) = I^{-2}(\theta_0) \cdot Var\left(U(\theta_0)\right)$$

By definition, the Fisher Information is:

$$Var\left(U(\theta_0)\right) = I(\theta_0)$$

Substituting this in, we get:

$$Var(\theta_k) = I^{-2}(\theta_0) \cdot I(\theta_0)$$

$$Var(\theta_k) = I^{-1}(\theta_0)$$

Therefore, the variance of the MLE θ_k is the inverse of the Fisher information.

Now coming back to the derivation of AIC, it is important to note that the following equation is in quadratic form.

$$\mathbb{E}[-L(y|\theta_k) + L(y|\theta_0)] = \mathbb{E}\left[-\frac{1}{2}(\theta_k - \theta_0)^T I(\theta_0)(\theta_k - \theta_0)^T\right]$$

We take out $-\frac{1}{2}$ out.

$$\mathbb{E}[-L(y|\theta_k) + L(y|\theta_0)] = -\frac{1}{2} \cdot \mathbb{E}[(\theta_k - \theta_0)^T I(\theta_0)(\theta_k - \theta_0)^T]$$

A quadratic form is an expression involving a vector and a matrix that results in a scalar. It is called a quadratic form because it involves terms that are quadratic (i.e., terms that involve squares and products of the vector components).

Mathematically, for a vector x and a symmetric matrix A, a quadratic form is given by:

$$Q(x) = x^T \cdot A \cdot x$$

In the context of parameter estimation and the AIC derivation, we consider the quadratic form involving the estimated parameter vector θ_k and the true parameter vector θ_0:

$$Q(\theta_k) = (\theta_k - \theta_0)^T I(\theta_0)(\theta_k - \theta_0)^T$$

Given that this in quadratic form, we can apply the generalized chi-distribution (Wikipedia).

$$X_k^2 \sim (\theta_k - \theta_0)^T I(\theta_0)(\theta_k - \theta_0)^T$$

Remember that a chi distribution has the following properties:

$$E[X_k^2] = k$$

$$Var[X_k^2] = 2k$$

And now remember that we had:

$$\mathbb{E}\left[d\left(\hat{\theta}_k\right)\right] = \mathbb{E}[-2\log\left(f\left(y|\hat{\theta}_k\right)\right] + [\mathbb{E}\left[d\left(\hat{\theta}_k\right)\right] - \mathbb{E}\left[-2\log(f(y|\theta_0)\right] + \mathbb{E}[-2\log(f(y|\theta_0)] - \mathbb{E}[-2\log\left(f\left(y|\hat{\theta}_k\right)\right]]$$

Given that we proved the two expressions = k, we get:

$$\mathbb{E}\left[d\left(\widehat{\theta_k}\right)\right] = -2 \cdot \log(L) + k$$

$$AIC = -2 \cdot \log(L) + k$$

This is derived similarly to Daniel F. Schmidt and Enes Makalic Model Selection with AIC Presentation.

9.9 Proof of BIC Criterion

Key assumptions made in the derivation, such as:

- Large sample size ($n \to \infty$).
- Regularity conditions for the likelihood function (e.g., differentiability).
- Positive definiteness of the Fisher Information Matrix for Gaussian Integral.

9.9.1 Laplace Approximation Rule

Firstly, we apply the Laplace Approximation rule.

We must first recall the Taylor Series Expansion as a function.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2!} + \cdots$$

Just note we are omitting higher-order terms in the Taylor expansion and the integral, mention that these terms become negligible for large n and do not significantly impact the approximation; however, it is included for mathematical sense.

If we assume our log-likelihood is differentiable, let us expand the log-likelihood function:

$$\log\left(p(y|\theta)\right) = \log\left(p\left(y|\hat{\theta}\right)\right) + \log\left(p\left(y|\hat{\theta}\right)\right)'\left(\theta - \hat{\theta}\right) + \frac{\log\left(p\left(y|\hat{\theta}\right)\right)''\left(\theta - \hat{\theta}\right)^2}{2!} + \cdots$$

Remember, we know:

$$\log\left(p\left(y|\hat{\theta}\right)\right)' = \mathbb{E}_\theta[S(\theta)] = \mathbb{E}_\theta\left[\frac{d}{d\theta}(\log[p(y|\theta)])\right] = 0$$

$\hat{\theta}$ is the MLE, so the gradient (first derivative) of the log-likelihood with respect to θ evaluated at $\hat{\theta}$ is zero. Note that θ is a vector of parameters of dimensions k.

Thus,

$$\log\left(p(y|\theta)\right) = \log\left(p\left(y|\hat{\theta}\right)\right) + \frac{\log\left(p\left(y|\hat{\theta}\right)\right)''\left(\theta - \hat{\theta}\right)^2}{2!} + \cdots$$

We can rearrange by taking $\frac{1}{2!} = \frac{1}{2}$

$$\log\left(p(y|\theta)\right) = \log\left(p\left(y|\hat{\theta}\right)\right) + \frac{1}{2}\log\left(p\left(y|\hat{\theta}\right)\right)''\left(\theta - \hat{\theta}\right)^2 + \cdots$$

The second derivative of the log-likelihood, $\log\left(p(y|\hat{\theta})\right)''$ represents the curvature of the log-likelihood function. This is considered the per observed Fisher Information.

$$\mathrm{F}(\hat{\theta}) = \log\left(p(y|\hat{\theta})\right)'' = -\frac{\partial^2}{\partial\theta^2}\log[p(y|\hat{\theta})]$$

Using this, we have now:

$$\log(p(y|\theta)) = \log\left(p(y|\hat{\theta})\right) - \frac{1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots$$

We apply the logarithmic, exponential rule:

$$e^{\log(x)} = x$$

Therefore,

$$\log(p(y|\theta)) = \log\left(p(y|\hat{\theta})\right) + \frac{-1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots$$

$$\exp[\log(p(y|\theta))] = \exp\left[\log\left(p(y|\hat{\theta})\right) + \frac{-1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right]$$

Which transforms to:

$$p(y|\theta) = \exp\left[\log\left(p(y|\hat{\theta})\right) + \frac{-1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right]$$

We apply the exponential rule:

$$e^{m+n} = e^m \cdot e^n$$

To get:

$$p(y|\theta) = \exp\left[\log\left(p(y|\hat{\theta})\right) + \frac{-1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right]$$

$$p(y|\theta) = \exp\left[\log\left(p(y|\hat{\theta})\right)\right] \cdot \exp\left[\frac{-1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right]$$

$$p(y|\theta) = p(y|\hat{\theta}) \cdot \exp\left[\frac{-1}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right]$$

9.9.2 Multivariate Gaussian Integral Rule

Secondly, we apply the Gaussian Integral rule.

Now given that the equation above is per observation for Fisher Information, The total Fisher Information for n independent observations is n times the per-observation Fisher Information because the information accumulates additively.

$$p(y|\theta) = p(y|\hat{\theta}) \cdot \exp\left[\frac{-n}{2}\mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right]$$

Before moving forward, we must understand the following content.
Marginal Likelihood is the probability of the observed data given a model, integrated over all possible values of the model's parameters. It quantifies how well the model M explains the observed data D, considering all possible values of the parameters θ.

$$p(y) = \int p(y|\theta) \cdot \pi(\theta)\, d\theta$$

$p(y|\theta)$ is the likelihood.
$\pi(\theta)$ is the prior distribution.

Note: We are considering a non-informative prior or that the prior does not depend significantly on θ in the region of interest.

Hence,

$$p(y|\theta) = \int p(y|\hat{\theta}) \cdot \exp\left[\frac{-n}{2} \mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right] \cdot \pi(\theta)\, d\theta$$

Given that $p(y|\hat{\theta})$ has variables of $\hat{\theta}$ and not θ, we can take it out of the integral.
We assume that $\pi(\theta)$ is a constant, we can take it out of the integral.

$$p(y|\theta) = p(y|\hat{\theta}) \cdot \pi(\theta) \int \exp\left[\frac{-n}{2} \mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right] d\theta$$

Before moving forward, we must understand the following content.

From the Wikipedia page on the Gaussian integral, we can extract the n-dimensional and functional generalisation multivariate Gaussian function formula over $\mathbb{R}^n$ is given by:

$$\int_{\mathbb{R}^n} \exp\left[-\frac{1}{2} x^T A x\right] d^n x = \sqrt{\frac{(2\pi)^n}{\det A}}$$

This is under the assumption that A is a symmetric positive-definite $n\ x\ n$ precision matrix, which is the matrix inverse of the covariance matrix.

We refer to the integral in reference:

$$I = \int \exp\left[\frac{-n}{2} \mathrm{F}(\hat{\theta})(\theta - \hat{\theta})^2 + \cdots\right] d\theta$$

From this we can see that $x = (\theta - \hat{\theta})^2$ and $A = n\mathrm{F}(\hat{\theta})$.
The dimension n in the integral corresponds to the number of parameters k.

$$I = \int_{\mathbb{R}^k} \exp\left[-\frac{1}{2} x^T \cdot n\mathrm{F}(\hat{\theta}) \cdot x\right] d^k x = \sqrt{\frac{(2\pi)^k}{\det[n\mathrm{F}(\hat{\theta})]}}$$

Using the following rules of determinants

$$\det[cA] = c^n \det[A]$$

Therefore,

$$\int_{\mathbb{R}^k} \exp\left[-\frac{1}{2} x^T \cdot n\mathrm{F}(\hat{\theta}) \cdot x\right] d^k x = \sqrt{\frac{(2\pi)^k}{n^k \cdot \det[\mathrm{F}(\hat{\theta})]}}$$

Substituting back into the integral, we get:

$$I = \sqrt{\frac{(2\pi)^k}{n^k \cdot \det[\mathrm{F}(\hat{\theta})]}} = \sqrt{\frac{(2\pi)^k}{n^k} \cdot \frac{1}{\det[\mathrm{F}(\hat{\theta})]}} = \sqrt{\frac{(2\pi)^k}{n^k} \cdot [\det \mathrm{F}(\hat{\theta})]^{-1}} = \sqrt{\frac{(2\pi)^k}{n^k}} \cdot \sqrt{[\det \mathrm{F}(\hat{\theta})]^{-1}} = \left(\frac{2\pi}{n}\right)^{\frac{k}{2}} \cdot (\det \mathrm{F}(\hat{\theta}))^{\frac{-1}{2}}$$

The equation is now:

$$p(y|\theta) = p(y|\hat{\theta}) \cdot \pi(\theta) \cdot \left(\frac{2\pi}{n}\right)^{\frac{k}{2}} \cdot (\det \mathrm{F}(\hat{\theta}))^{\frac{-1}{2}}$$

Take the natural log for both sides

$$\log[p(y|\theta)] = \log[p(y|\hat{\theta}) \cdot \pi(\theta) \cdot \left(\frac{2\pi}{n}\right)^{\frac{k}{2}} \cdot (\det \mathrm{F}(\hat{\theta}))^{\frac{-1}{2}}]$$

We apply the log law:

$$\log M \cdot N = \log M + \log N$$

To get:

$$\log[p(y|\theta)] = \log[p(y|\hat{\theta})] + \log[\pi(\theta)] + \log[\left(\frac{2\pi}{n}\right)^{\frac{k}{2}}] + \log[\left(\det \mathrm{F}(\hat{\theta})\right)^{\frac{-1}{2}}]$$

Apply the log law:

$$\log m^k = k \cdot \log\,(m)$$

$$\log[p(y|\theta)] = \log[p(y|\hat{\theta})] + \log[\pi(\theta)] + \frac{k}{2}\log[\frac{2\pi}{n}] - \frac{1}{2}\log[\det \mathrm{F}(\hat{\theta})]$$

For large n, the terms $\det \mathrm{F}(\hat{\theta})$ and $\pi(\theta)$ are $O(1)$, so we can focus on the leading terms.

$$\log[p(y|\theta)] = \log[p(y|\hat{\theta})] + \frac{k}{2}\log[\frac{2\pi}{n}]$$

We simplify using the log law:

$$\log \frac{M}{N} = \log M - \log N$$

To get:

$$\log[p(y|\theta)] = \log[p(y|\hat{\theta})] + \frac{k}{2}\log[2\pi] - \frac{k}{2}\log[n]$$

We recognise that $\frac{k}{2}\log[2\pi]$ is a constant term, thus,

$$\log[p(y|\theta)] = \log[p(y|\hat{\theta})] - \frac{k}{2}\log[n]$$

Multiply both sides by -2

$$-2\log[p(y|\theta)] = -2\log[p(y|\hat{\theta})] + k\log[n]$$

Thus, BIC is:

$$BIC = -2\log[p(y|\hat{\theta})] + k\log[n]$$

---

10. APPLICATION TO TIME SERIES MODELS

10.1 Autocorrelation Function (ACF)

Time series analysis is all about modelling and forecasting a series based on the autocorrelation, a measure for the link between the present and the past.
Suppose the y series is stationary and denote its mean value by μ.

Sample Mean ($\bar{y}$):
The sample mean is the average value of the time series data. In this context, it is the average closing stock price over the given days. It provides a central value around which each day's price varies.

Variance ($\sigma_0{}^2$) – Population Variance:
Variance measures the spread of the stock prices around their mean. It's a measure of the overall variability of the series. In this example, the variance gives you an idea of how much the closing prices fluctuate on average from the mean price.

Autocovariance ($\sigma_k{}^2$):
This measures how much two random variables (separated by k periods) change together. It is the covariance of the series with its own lagged values.
For $\sigma_1{}^2$, a positive autocovariance implies that if a stock price is above the mean on one day, it is likely to be above the mean the following day as well. A negative autocovariance, as in this example, suggests that prices tend to move in opposite directions from one day to the next. *The value is arbitrary. Autocorrelation determines the numerical, linear relationship.*

Autocorrelation (γ_k)
Autocorrelation, calculated from the autocovariance. It normalises the covariance ($\sigma_k{}^2$) by the variance.
It tells you the strength and direction of a linear relationship between time series observations at different times.
It is a value that can range from -1 to +1.
Here, an autocorrelation of 0.625 suggests a moderate positive relationship between consecutive days: if the price is higher than average today, it is somewhat likely to be higher than average tomorrow as well.

Sample Autocorrelation ($\hat{\gamma}_1$)
Basically the same as γ_k, just assuming we are working with a sample.
This is the estimate of the autocorrelation calculated from sample data.
It is used to infer the correlation in the overall population of data (beyond just the sample).

10.2 Partial Autocorrelation Function (PACF)

The Partial Autocorrelation Function (PACF) is a more nuanced measure used in time series analysis compared to the basic autocorrelation function (ACF). The PACF measures the correlation between a series and its lag, like the ACF, but it goes further by excluding the influence of the correlations of shorter lags.

Autocorrelation Function (ACF) for lag 2 measures the correlation between today's stock price and the stock price two days ago, but this correlation includes all the indirect effects through the stock price from one day ago. Essentially, it doesn't isolate the direct impact of the stock price from two days ago; it just measures the overall correlation.

Partial Autocorrelation Function (PACF) for lag 2, on the other hand, specifically measures the direct correlation between today's stock price and the stock price two days ago, after removing or controlling for the effects of the stock price from one day ago. It isolates the influence of the stock price two days ago from any intervening effects.

Given a time series z_t, the partial autocorrelation of lag k, denoted by $\rho_{k,k}$, is the autocorrelation between z_t and z_{t+k} with linear dependence on z_t and z_{t+1} through z_{t+k-1}.

$$\rho_{1,1} = corr(x_{t+k}, x_t)$$

For k = 1. This represents the partial autocorrelation (PACF) at lag 1

$$\rho_{1,1} = corr(x_{t+1}, x_t)$$

Why is the subscript k, k and not just k?

- The first k represents the lag, indicating that we are considering the correlation between z_t and z_{t-k}.
- The second k highlights that this correlation has been adjusted by removing the influence of all intermediate values between t and t−k, specifically the values from t−1 to t−(k−1).

Model for Lag 1

$$Y_t = \rho_{11} Y_{t-1} + u_t$$

Here, we regress Y_t directly against Y_{t-1} (only considering the immediate previous day).

Model for Lag 2

$$Y_t = \rho_{21} Y_{t-1} + \rho_{22} Y_{t-2} + u_t$$

This model checks the influence of the price two days ago on today's price, controlling for the effect of just the previous day.

---

11. MINIMUM MESSAGE LENGTH

11.1 Introduction to MML

The Minimum Message Length (MML) framework provides an information-theoretic approach to statistical inference, aiming to find a balance between model complexity and goodness of fit. It combines the principles of Bayesian inference and information theory to identify the model that minimises the total message length required to encode both the model and the data it explains. This method has been particularly valuable for model selection, parameter estimation, and hypothesis testing across a wide range of applications.

MML divides the encoding process into two parts:

(1) Model Complexity ($I(\theta)$):

This represents the length of the message needed to describe the model, including all its parameters.
A simpler model (fewer or more general parameters) results in a shorter encoding for this part.
Encourages adherence to Occam's Razor, favoring simpler models unless the added complexity significantly improves fit.

(2) Goodness of Fit ($I(y|\theta)$):
This quantifies the length of the message required to describe the data, assuming the model is correct.
A better-fitting model reduces this part by requiring fewer bits to encode the residuals or deviations between observed and predicted data.

The total message length, represented as:

$$\text{msgLen}(\beta \wedge D) = I(\theta) + I(y|\theta)$$

guides model selection by balancing these two components.

The key insight in MML is the quantitative trade-off:

- Reducing $I(\theta)$: Achieved by using simpler models with fewer parameters. However, oversimplified models may fail to capture the data's complexity, increasing.
- Reducing $I(y|\theta)$: Achieved by fitting the data more closely, which may involve more complex models. Overfitting risks inflating.

The optimal model minimises the total message length, ensuring it is both compact (less complex) and descriptive (well-fitted to the data).

MML aims to solve:

$$\theta^* = \arg\max_{\theta \in \Theta} \{I(\theta) + I(y|\theta)\}$$

where:

- θ^*: Optimal parameter set.
- Θ: Space of all possible parameter sets.

11.2 Shannon's Information Theory

MML is rooted in Shannon's Information Theory, where the length of a message encoding an event is proportional to its negative logarithmic probability:

$$I(E) = -\log(\text{pr}(E))$$

This ensures rare events carry more "information," reflecting their surprising nature.

11.3 Bayes' Theorem

$$pr(H|D) = \frac{pr(D|H) \cdot pr(H)}{pr(D)}$$

This theorem updates the probability of a hypothesis H given new data D.

11.4 Bayesian Inference

MML incorporates Bayesian inference, combining:

Prior Distribution (h(θ)): Encodes prior beliefs about the parameters, influencing $I(\theta)$.
Likelihood Function (f(y|θ)): Encodes the probability of observing data y given θ, influencing $I(y \mid \theta)$.

By integrating these components, MML provides a rigorous method for selecting models that balance prior knowledge and data-driven insights.

11.5 Bayesian Prior Distribution

The Bayesian prior distribution plays a crucial role in the Minimum Message Length (MML) framework, which combines Bayesian inference and Shannon's information theory to form an encoding-based approach for model selection and parameter estimation. The MML approach considers both the encoding of the model parameters (prior

knowledge) and the data given those parameters to achieve a concise description, aligning well with Bayesian reasoning by integrating prior beliefs and observed data to update parameter estimates.

11.5.1 Uniform Prior Distribution ($h_3(\beta)$)

To simplify calculations in our analysis, we assume a uniform prior distribution for the model parameters, denoted $h_3(\beta) = 1$. This uniform prior implies that any value of β between the specified bounds [0, 1] is equally probable, allowing for a straightforward approach to Bayesian encoding within MML. By using this prior, we assign an equal likelihood to all parameter values within the range, enabling us to focus on encoding the parameters and data efficiently without additional adjustments for varying prior probabilities.

As per the report: "we model the parameter set β using uniform prior [0, 1] in the stationarity region $h_3(\beta) = 1$".

11.6 MML First Component

$$msglen(\beta) = -\log[h_3(\beta) \cdot s]$$

Note we assume β contains all the coefficients from ARMA model.

$$\beta = (\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q, \sigma^2)$$

where $h_3(\beta)$ is the prior distribution over the parameter β.

11.6.1 What is $-log(p)$?

Shannon's information theory states that the amount of information (in bits) associated with an event of probability p.

Shannon's information theory provides a way to quantify the amount of information carried by an event. The key idea is that rare events (low probability) carry more information than common events (high probability). This is because when a rare event occurs, it is more surprising and thus provides more "new" information.

$$-log(p)$$

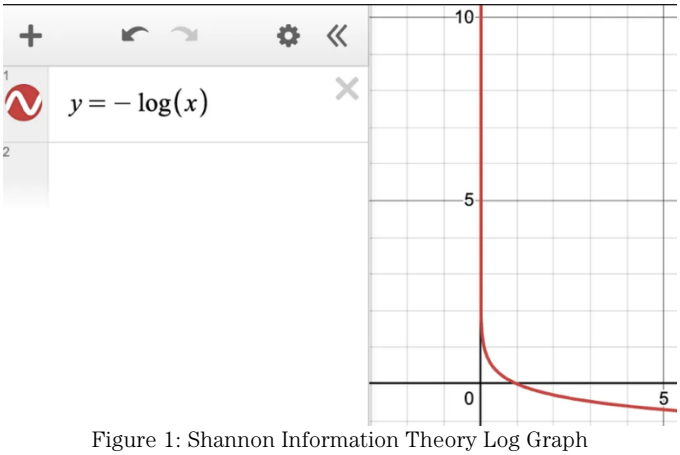


Figure 1: Shannon Information Theory Log Graph

High Probability (p≈1): Common events carry little information.
For example, if p=1 (100% of event happening), then I(E) = −log(1) = 0 bits.

Low Probability (p<1): Rare events carry a lot of information.
For example, if p=1/1000, then I(E) = −log(1/1000) ≈ 10 bits.

11.6.2 What is s?

Precision 's'

In the context of MML, 's' represents the precision with which we can measure or encode a parameter θ. When we say we have a precision of 's', it means that our estimate of θ is not exact but falls within a small interval of width 's'.

Example of Precision

- Suppose we are measuring the length of a table, which we can measure to the nearest centimetre. If we measure the length to be 150 cm, we mean the true length is somewhere between 149.5 cm and 150.5 cm. Here, the precision 's' is 1 cm.

For better approximation for the value of s and why it is the following, look below:

$$s = \sqrt{\frac{12}{F(B)}}$$

11.6.3 What is t?

Letting $\theta' = \theta + t$

This notation means we are considering small deviations t from the parameter θ within the interval of precision.

$$t = -\frac{s}{2} < t < +\frac{s}{2}$$

Interval Example

- Suppose θ is estimated as 150 cm, and the precision 's' is 1 cm. Then, θ′ can vary within the interval 149.5 ≤ θ′≤ 150.5.

11.6.4 Derivation of First MML Component

$$msglen(\beta) = -\log[h_3(\beta) \cdot s]$$

Substitute s (derived below in 11.7) into $msgLen(B)$

$$msgLen(\beta) = -\log\left(h_3(\beta) \cdot \sqrt{\frac{12}{F(\beta)}}\right)$$

Apply multiplication log law

$$msgLen(\beta) = -\left[\log\big(h_3(\beta)\big) + log\left(\sqrt{\frac{12}{F(\beta)}}\right)\right]$$

Expand negative

$$msgLen(\beta) = -\log\big(h_3(\beta)\big) - log\left(\sqrt{\frac{12}{F(\beta)}}\right)$$

Take out square root using log laws

$$msgLen(\beta) = -\log\big(h_3(\beta)\big) - \frac{1}{2}\log\left(\frac{12}{F(\beta)}\right)$$

Adjust division using log laws

$$msgLen(\beta) = -\log\big(h_3(\beta)\big) - \frac{1}{2}\big[\log(12) - \log\big(F(\beta)\big)\big]$$

Expand ½

$$msgLen(\beta) = -\log\big(h_3(\beta)\big) - \frac{1}{2}\log(12) + \frac{1}{2}\log\big(F(\beta)\big)$$

Take $\frac{1}{2}log(12)$ as constant, we get:

$$msgLen(\beta) = -\log\big(h_3(\beta)\big) + \frac{1}{2}\log\big(F(\beta)\big) + constant$$

11.6.5 Lattice Constant

To account for the discretisation of the parameter space, we include lattice constants κ_k, which adjust for the expected error due to quantisation. The lattice constants depend on the number of parameters k.

The term $(\frac{k}{2}) * log(\kappa_k)$ accounts for this discretisation.

Combining these terms, we get the total message length for encoding the model:

$$msgLen(\beta) = -\log(h_3(\beta)) + \frac{1}{2}\log(F(\beta)) + \frac{k}{2}\log(\kappa_k) + \frac{k}{2}$$

11.7 MML Second Component

$$msgLen(\beta \wedge D) = msgLen(\beta) + msgLen(D \mid \beta)$$

msgLength of data given the parameters can also be written as:

$$MessLen_{data} = -\log[f(y \mid \beta)]$$

where $f(y \mid \beta)$ is the likelihood of the data given the parameters.

From information theory, the optimal precision s is related to the Fisher Information by:

$$s = \sqrt{\frac{12}{F(B)}}$$

For more information on what $F(B)$ please reference section 9.7.

11.7.1 Derivation of 's'

What is s?

In the context of MML, s represents the precision with which we encode the parameter β.
The idea is to determine the optimal width s of the interval around β such that the total message length is minimised.
Below is a step-by-step explanation on the derivation.

1. Taylor Expansion of Log-Likelihood

To find the optimal precision s, we use a Taylor series expansion of the log-likelihood function around the parameter β.

Assume we have a parameter θ and we want to approximate the log-likelihood function $-logf(x \mid \theta)$ around θ using a small deviation t:

$$\theta' = \theta + t \quad \text{where } -\frac{s}{2} \leq t \leq \frac{s}{2}$$

Using a Taylor series expansion, we can approximate $-logf(x \mid \theta + t)$. Please reference Appendices 16.1 for more context.

The Taylor series of a function f(x) around the point x=a is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

Here:

- x is the variable of the function.
- a is the point around which the expansion is done.

In the context of the problem:

- x: The variable we are considering the change in, which in this case is $\theta + t$.
    - Like in $f(x) = e^x$, x is x for the function.
    - In this case, $\theta + t$ is our x for the function ($x = \theta + t$).

- a: The point around which we are expanding, which is θ.
    - Like in $f(x) = e^x$, $a = 0$ (this is the point we are trying to approximate around)

Thus, when substituting the values:

$$f(\theta + t) = f(\theta) + f'(\theta)(\theta + t - \theta) + \frac{f''(\theta)}{2!}(\theta + t - \theta)^2 + \frac{f'''(\theta)}{3!}(\theta + t - \theta)^3 + \cdots$$

Simplifying this we get:

$$f(\theta + t) = f(\theta) + f'(\theta)(t) + \frac{f''(\theta)}{2!}(t)^2 + \frac{f'''(\theta)}{3!}(t)^3 + \cdots$$

$$f(\theta + t) = f(\theta) + f'(\theta)(t) + \frac{f''(\theta) \cdot t^2}{2!} + \frac{f'''(\theta) \cdot t^3}{3!} + \cdots$$

Let's consider an example with the function $f(x) = -logf(x|\theta + t)$:

$$f(\theta) = -log\, f(x|\theta)$$

$$f'(\theta) = \frac{d}{d\theta}\big(-log\, f(x|\theta)\big)$$

$$f''(\theta) = \frac{d^2}{d\theta^2}\big(-log\, f(x|\theta)\big)$$

$$-log\, f(x|\theta + t) = -log\, f(x|\theta) + \left(\frac{d}{d\theta}\big(-log\, f(x|\theta)\big)\right)(t) + \left(\frac{d^2}{d\theta^2}\big(-log\, f(x|\theta)\big)\right)\frac{t^2}{2!}$$

Note: Higher-order terms (t^3 and above) are neglected because t is small.

2. Averaging Over The Interval

The idea here is to understand how the function $-logf(x \mid \theta + t)$ behaves not just at a single point, but averaged over a small interval around θ. This can be particularly useful in statistical contexts where we are interested in the behavior of the likelihood function over a range of parameter values, rather than at a single point.

Remember: t is the number of small deviations from the parameter θ within the interval of precision.

We want to average this expression over an interval:

$$t = -\frac{s}{2} < t < +\frac{s}{2}$$

$$t \in \left(-\frac{s}{2}, +\frac{s}{2}\right)$$

Why Averaging?

Averaging over an interval provides insight into the general behaviour of the function around the point θ.
It's a way to smooth out the function to understand its average behaviour within a specific range.
The linear term:

$$t\left(\frac{d}{d\theta}\big(-log\, f(x|\theta)\big)\right)$$

When averaging over the interval $t \in (-\frac{s}{2}, \frac{s}{2})$, the linear term averages to zero. This is because t is symmetric around zero and the positive and negative values of t cancel each other out. Mathematically:

$$\frac{1}{s} \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} t^2\, dt = \frac{1}{s} \cdot 0 = 0$$

Quadratic Term

$$\frac{t^2}{2!}\left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right)$$

The quadratic term does not average to zero. Instead, we need to calculate the average of t^2 over the interval $t \in (-\frac{s}{2}, \frac{s}{2})$.

The average of t^2 over the interval $t \in (-\frac{s}{2}, \frac{s}{2})$ is given by:

$$\int_{-\frac{s}{2}}^{\frac{s}{2}} t^2\ dt = \left[\frac{t^3}{3}\right]_{-\frac{s}{2}}^{\frac{s}{2}}$$

$$= \frac{\left(\frac{s}{2}\right)^3}{3} - \frac{\left(-\frac{s}{2}\right)^3}{3}$$

$$= \frac{\frac{s^3}{2^3}}{3} - \frac{\frac{-s^3}{2^3}}{3} = \frac{s^3}{2^3 \cdot 3} + \frac{s^3}{2^3 \cdot 3} = \frac{s^3}{24} + \frac{s^3}{24} = \frac{2 \cdot s^3}{24} = \frac{s^3}{12}$$

3. Message Length for the Data Given Parameter

The second part of the message length, which is the message length for the data given the parameter, can be approximated as.

We use the Average Value Theorem:

$$\frac{1}{b-a}\int_a^b f(x)\ dx$$

Average Value Theorem

- The integral of t^2 from $-\frac{s}{2}$ to $\frac{s}{2}$ gives us the total sum of t^2 over this interval, which in the context of our problem represents the total impact of the quadratic term in the Taylor expansion.
- To find the average impact per unit length (i.e., how much each small section of the interval contributes on average), we divide this total by the length of the interval, s. This division converts a total sum into an average value per unit interval.
- Conceptually, this is similar to finding the average value of a function over an interval, where you want to know the typical value of the function rather than just the total accumulated effect.

$$\frac{1}{\left(\frac{s}{2}\right) - \left(\frac{-s}{2}\right)} = \frac{1}{s}$$

$$\frac{1}{s} \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} t\ dt\ = \frac{1}{s} \cdot\ 0 = 0$$

Thus, from this equation

$$-log\ f(x|\theta + t) = -log\ f(x|\theta) + \left(\frac{d}{d\theta}(-log\ f(x|\theta))\right)(t) + \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right)\frac{t^2}{2!}$$

The following changes are made to get the average over the interval of t.
Linear Term:

$$t\ \left(\frac{d}{d\theta}(-log\ f(x|\theta))\right)$$

$$\frac{1}{s} \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} \left[t \cdot \left(\frac{d}{d\theta}(-\log f(x|\theta))\right)\right] dt$$

Since $(\frac{d}{d\theta}(-\log f(x|\theta))$ is constant with respect to t, we can factor it out of the integral:

$$\frac{1}{s} \cdot \left(\frac{d}{d\theta}(-\log f(x|\theta)) \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} t \; dt\right.$$

$$\frac{1}{s} \cdot \left(\frac{d}{d\theta}(-\log f(x|\theta)) \cdot 0 = 0\right.$$

The quadratic term

$$\frac{1}{s} \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} \left[\frac{t^2}{2!}\left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right)\right] dt$$

Since $\left(\frac{d^2}{d\theta^2}(-\log f(x|\theta)\right)$ is constant with respect to t, we can factor it out of the integral:

$$\frac{1}{s} \cdot \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} \frac{t^2}{2!}\ dt$$

Same with $\frac{1}{2!}$

$$\frac{1}{s} \cdot \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{1}{2!} \cdot \int_{-\frac{s}{2}}^{\frac{s}{2}} t^2\ dt$$

Remember:

$$\int_{-\frac{s}{2}}^{\frac{s}{2}} t^2\ dt = \frac{s^3}{12}$$

Thus,

$$\frac{1}{s} \cdot \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{1}{2!} \cdot \frac{s^3}{12}$$

$$\left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{s^2}{24}$$

So now the main equation here:

$$-log\ f(x|\theta + t) = -log\ f(x|\theta) + \left(\frac{d}{d\theta}(-log\ f(x|\theta))\right)(t) + \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right)\frac{t^2}{2!}$$

We replace the linear term and the quadratic term to get:

$$-log\ f(x|\theta + t) = -log\ f(x|\theta) + 0 + \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{s^2}{24}$$

$$-log\ f(x|\theta + t) = -log\ f(x|\theta) + \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{s^2}{24}$$

We combine both parts of the MML equation:

$$msgLen(\theta) = -log(h(\theta) \cdot s)$$

$$msgLen(D|\theta) = -log\ f(x|\theta) + \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{s^2}{24}$$

Combining these:

$$Total\ msgLen = -log(h(\theta) \cdot s) - log\ f(x|\theta) + \left(\frac{d^2}{d\theta^2}(-log\ f(x|\theta))\right) \cdot \frac{s^2}{24}$$

To find the optimal s, we minimise the total message length. Differentiate the total message length with respect to s and set it to zero:

$$\frac{\partial}{\partial s}\left[-\log(h(\theta)\cdot s\,)-\log\big(f(x|\theta)\big)+\frac{s^2}{24}\left(\frac{d^2}{d\theta^2}\big(-\log\big(f(x|\theta)\big)\big)\right)\right]\;=\;0$$

Solving this gives us:

$$\frac{\partial}{\partial s}\left[-\log(s\,)+\frac{s^2}{24}\left(\frac{d^2}{d\theta^2}\big(-\log\big(f(x|\theta)\big)\big)\right)\right]\;=\;0$$

Note: $h(\beta)$ and $f(y|\beta)$ do not depend on s, so their derivatives with respect to s are zero.
Remember: $\frac{d^2}{d\theta^2}\big(-\log\big(f(x|\theta)\big)\big)$ is a constant as defined above.

This simplifies to:

The differentiation rules are applied for the first term:

$$\frac{d}{dx}[\log(x)]=\frac{1}{x}$$

The differentiation rules are applied for the second term is just normal basic rules.

$$2\cdot\frac{s^2}{4}=\frac{1}{2}\cdot 2\cdot 2$$

Thus, we get:

$$-\frac{1}{\mathrm{s}}+\frac{s}{12}\left(\frac{d^2}{d\theta^2}\big(-\log\big(f(x|\theta)\big)\big)\right)=\;0$$

Multiplying both sides by s to simplify:

$$-1+\frac{s^2}{12}\left(\frac{d^2}{d\theta^2}\big(-\log\big(f(x|\theta)\big)\big)\right)=\;0$$

We also know that Fisher Information is equal to:

$$F(\theta)=\frac{d^2}{d\theta^2}\big(-\log\big(f(x|\theta)\big)\big)$$

Solving for s:

$$-1+\frac{s^2}{12}\cdot F(\theta)=\;0$$

$$\frac{s^2}{12}\cdot F(\theta)=\;1$$

$$s^2\cdot F(\theta)=\;12$$

$$s^2=\frac{12}{F(\theta)}$$

$$s=\sqrt{\frac{12}{F(\theta)}}$$

11.8 Derivation of MML

Now putting 11.6 and 11.7 together, we get:

$$msgLen(\beta\wedge D)=msgLen(\beta)+msgLen(\,D\mid\beta\,)$$

$$msgLen(\beta\wedge D)=\left[-\log\big(h_3(\beta)\big)+\frac{1}{2}\log\big(F(\beta)\big)+\frac{k}{2}\log(\kappa_k)+\frac{k}{2}\right]+\left[-\log\big(f(y|\beta)\big)\right]$$

$$msgLen(\beta \wedge D) = -\log\big(h_3(\beta)\big) + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}\log(\kappa_k) + \frac{k}{2} + -\log\big(f(y|\beta)\big)$$

11.9 Report Proof From Equation (8) to Equation (9)

$$MessLen(y,\beta) = -\log\left(\frac{h_3(\beta)f(y_1,\ldots,y_N\beta)\epsilon^N}{\sqrt{F(\beta)}}\right) + \frac{k}{2}(1+\log\kappa_k) - \log h_1(p) - \log h_2(q), \quad (8)$$

where ϵ is measuring the accuracy of data, $h_3(\beta)$ is the Bayesian prior distribution over the parameter set β, we model the parameter set β using uniform prior [0, 1] in the stationarity region $h_3(\beta) = 1$, and $h_1(p) = 2^{-(1+p)}$ and $h_2(q) = 2^{-(1+q)}$ are the priors on the (non-negative integer) parameters p and q, $k = p + q + 1$ is the number of continuous-valued parameters, $f(y_1,\ldots,y_N\beta)$ is the standard statistical likelihood function, $L = -\log f$, $F(\beta)$ is the expected Fisher Information matrix (of expected second-order partial derivatives of L) and is a function of the parameter set β, $F(\beta)$ is the expected Fisher information, κ_k is the lattice constant (which accounts for the expected error in the log-likelihood function from ARMA model (Equation (6)) due to the quantization of the k-dimensional space, which is bounded above by $\frac{1}{12}$ and bounded below by $\frac{1}{2\pi e}$. For example, $\kappa_1 = \frac{1}{12}$, $\kappa_2 = \frac{5}{36\sqrt{3}}$, $\kappa_3 = \frac{19}{192*2^{1/3}}$, and $\kappa_k \to \frac{1}{2\pi e}$ as $k \to \infty$).

Ignoring the $-\log h_1(p)$, $-\log h_2(q)$, and $-N\log(\epsilon)$ terms, the message length for the ARMA model β can also be represented as:

$$I(y,\beta) = -\log h_3(\beta) + \frac{1}{2}\log F(\beta) + \frac{k}{2}\log\kappa_k + \frac{k}{2} - \log f(y\beta) \quad (9)$$

Figure 2: "Minimum Message Length in Hybrid ARMA and LSTM Model Forecasting" Minimum Message Length Section

$$MessLen(y,\beta) = -\log\left(\frac{h_3(\beta)\cdot f(y_1,\ldots,y_N)\varepsilon^N}{\sqrt{F(\beta)}}\right) + \frac{k}{2}(1+\log(\kappa_k) - \log(h_1\cdot p) - \log(h_2\cdot q))$$

We apply the quotient rule

$$\log\left(\frac{x}{y}\right) = \log(x) - \log(y)$$

$$MessLen(y,\beta) = -\left[\log(h_3(\beta)\cdot f(y_1,\ldots,y_N)\varepsilon^N) - \log\left(\sqrt{F(\beta)}\right)\right] + \frac{k}{2}(1+\log(\kappa_k)) - \log(h_1\cdot p) - \log(h_2\cdot q)$$

$$MessLen(y,\beta) = -\log(h_3(\beta)\cdot f(y_1,\ldots,y_N)\varepsilon^N) + \log\left(\sqrt{F(\beta)}\right) + \frac{k}{2}(1+\log(\kappa_k)) - \log(h_1\cdot p) - \log(h_2\cdot q)$$

We apply the power log rule

$$\log(M^k) = k\cdot\log(M)$$

$$MessLen(y,\beta) = -\log(h_3(\beta)\cdot f(y_1,\ldots,y_N)\varepsilon^N) + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}(1+\log(\kappa_k)) - \log(h_1\cdot p) - \log(h_2\cdot q)$$

We apply the multiplication log rule

$$\log(\mathrm{M}\cdot\mathrm{N}) = \log(M) + \log(N)$$

To make it easier, we are going to take out a negative

$$MessLen(y,\beta) = -\left[\log\big(h_3(\beta)\big) + \log\big(f(y_1,\ldots,y_N)\big) + \log(\varepsilon^N)\right] + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}(1+\log(\kappa_k)) - \log(h_1\cdot p) - \log(h_2\cdot q)$$

$$MessLen(y,\beta) = -\log\big(h_3(\beta)\big) - \log\big(f(y_1,\ldots,y_N)\big) - \log(\varepsilon^N) + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}(1+\log(\kappa_k)) - \log(h_1\cdot p) - \log(h_2\cdot q)$$

We apply the power log rule for $\log(\varepsilon^N)$

$$MessLen(y,\beta) = -\log\big(h_3(\beta)\big) - \log\big(f(y_1,\ldots,y_N)\big) - \mathrm{N}\log(\varepsilon) + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}(1+\log(\kappa_k)) - \log(h_1\cdot p) - \log(h_2\cdot q)$$

We remove $\log(h_1\cdot p)$, $\log(h_2\cdot q)$ and $\mathrm{N}\log(\varepsilon)$

- $\log(h_1\cdot p)$ and $\log(h_2\cdot q)$ terms are related to the priors on specific parameters p and q of the ARMA model. If these parameters are assumed to be known or constant, or if the impact of their priors is negligible

compared to other components of the model, these terms can be removed. This is often done to simplify calculations without significantly affecting the outcome, especially if these priors do not vary much or if they are uniform.

- $\text{Nlog}(\varepsilon)$ involves ϵ, which is often related to the noise or error term in the model. If ϵ is considered constant or its log term is small relative to other terms, it might be dropped to focus on more significant components of the message length. This assumption might hold if the noise is uniform or well-regulated across the data set.

$$MessLen(y,\beta) = -\log\big(h_3(\beta)\big) - \log\big(f(y_1,\dots,y_N)\big) + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}(1+\log(\kappa_k))$$

Rearrange around

$$MessLen(y,\beta) = -\log\big(h_3(\beta)\big) + \frac{1}{2}\log\big(F(\beta)\big) - \log\big(f(y_1,\dots,y_N)\big) + \frac{k}{2}(1+\log(\kappa_k))$$

Expand the $\frac{k}{2}$ bracket

$$MessLen(y,\beta) = -\log\big(h_3(\beta)\big) + \frac{1}{2}\log\big(F(\beta)\big) - \log\big(f(y_1,\dots,y_N)\big) + \frac{k}{2} + \frac{k}{2}\log(\kappa_k)$$

Rearrange again

$$MessLen(y,\beta) = -\log\big(h_3(\beta)\big) + \frac{1}{2}\log\big(F(\beta)\big) + \frac{k}{2}\log(\kappa_k) + \frac{k}{2} - \log\big(f(y_1,\dots,y_N)\big)$$

11.10 Additional Information on MML and Fisher Information

Discrete Minimum Message Length

Minimum Message Length (MML) estimation is a Bayesian framework that integrates model selection and parameter estimation by considering the precision of measurements. Introduced in 1968, MML has been applied to various statistical distributions and is notable for using the Fisher information to optimally discretise continuous parameters.

This utilisation ensures that the posterior distribution remains a true probability rather than a probability density, which is essential for coherent statistical inference.

In our context, we are using a uniform prior distribution for simplicity. The general form of MML, often referred to as "MML87," depends on the determinant of the Fisher information matrix. The Fisher information quantifies the amount of information that an observable random variable carries about an unknown parameter, thus playing a critical role in determining the optimal precision with which to encode parameters.

Strict Minimum Message Length (SMML)

Strict Minimum Message Length (SMML) inference, developed by Wallace and Boulton in 1975, constructs a mapping from the data space to the set of models (parameters) to minimise the expected length of a two-part message comprising the model and the data given the model. This mapping defines a partition of the data space, where each part corresponds to data values that map to a specific parameter value. SMML estimators are optimal in the sense that they are invariant, consistent, and effectively handle model selection, parameter estimation, and hypothesis testing.

However, SMML inference is NP-hard for most problems, making it computationally infeasible in general cases. While a polynomial-time algorithm exists for specific distributions like the Binomial distribution, the complexity remains prohibitive for broader applications. To overcome this limitation, the MML method (as outlined by Wallace and Boulton in 1968 and refined by Wallace and Freeman in 1987), “MML87”, serves as a feasible approximation to SMML. MML makes simplifying approximations that render the computations tractable while still providing robust statistical inference.

It's important to note that while SMML does not incorporate these simplifying approximations, the mathematical and algorithmic challenges it poses can be significant. Therefore, MML offers a practical balance between computational efficiency and statistical rigor. The reliance on the Fisher information matrix in MML highlights its fundamental role in capturing the essence of the data with respect to the parameters being estimated.

12. DEEP LEARNING ARCHITECTURES

12.1 Introduction

To understand Long Short-Term Memory (LSTM) networks, it is essential to first analyse Recurrent Neural Networks (RNNs) in detail, as LSTMs are an extension designed to address specific limitations inherent to RNNs. This section provides a breakdown of RNNs, explaining their architecture, functionality, and the mathematical foundations underlying their operation, including the intricacies of backpropagation through time (BPTT). We will also examine the challenges faced by RNNs, such as vanishing and exploding gradients, which are unable to capture long-term dependencies effectively.

Building upon this understanding, we will delve into LSTMs, exploring how they mitigate these limitations through a carefully designed gating mechanism. This section also includes an in-depth mathematical analysis of LSTM operations and the backpropagation process that enables their learning. By the end of this section, the reader will have a thorough understanding of both architectures, their mechanics, and the foundational mathematics driving their performance.

12.2 Recurrent Neural Networks (RNN)

12.2.1 Background

To get an understanding on LSTMs, we must first cover the foundations of RNNs.

Modelling time-series data and predicting future values using various probabilistic models such as Autoregressive models (AR). These models are used to analyse sequential or time-dependent data by utilising past observations to predict future values.

Unlike AR, RNNs do not incorporate a probabilistic component into their structure. This means that while models like AR explicitly model uncertainty and probability distributions of states and observations, RNNs do not have built-in mechanisms to handle probabilities directly.

The primary goal of an RNN is to learn the relationships (associations) between inputs and targets. In RNNs, the hidden states are not modelled with any probabilistic assumption. They are deterministic given the previous state and the current input. This makes RNNs fundamentally different in how they process and generate data, focusing more on learning patterns and sequences directly from the data without modelling the underlying data generation process probabilistically.

RNNs transform inputs into real-valued vector representations that capture the information necessary for predicting or classifying sequences. These vector representations are learned and adjusted during training to optimise performance on the given task.

Input Vector (x_t): This is the real-valued vector that represents the input data at time step t. It could be a segment of a sequence like a frame in a video, a word in text processing, or a time point in a time-series analysis.

Hidden State (h_t): This is the internal memory of the RNN, which is updated at each time step. It carries information from previous time steps to the current step and is crucial for capturing temporal dependencies in the data.

Output Vector ($\hat{y}$): This is the predicted output of the RNN, computed from the current hidden state. It can be a class label, a future value in a series, or any other type of prediction relevant to the task.

Hidden State Update Equation

$$h_t = \tanh(Ux_t + Wh_{t-1})$$

- U and W are weight matrices.
    - U connects the input vector x_t to the hidden state
    - W connects the previous hidden state h_{t-1} to the current one.
- tanh is the activation function that introduces non-linearity into the model and helps to control the range of the hidden state values, keeping them between -1 and 1. This is important for managing gradients during learning and avoiding problems like exploding gradients.

U (Input-Hidden Matrix): This matrix is responsible for transforming the input vector x_t at each time step into a suitable form to be processed by the hidden layer. It maps the input data into the hidden space.

W (Hidden-Hidden Matrix): This matrix connects the hidden state from the previous time step (h_{t-1}) to the current hidden state (h_t). It is crucial for capturing temporal dependencies in the data, as it allows the network to "remember" previous information.

V (Hidden-Output Matrix): This matrix transforms the hidden state (h_t) into the output vector ($\hat{y}$). It shapes the final output of the network.

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

The tanh function then acts on the linear combination of these transformations to produce the new hidden state, which includes the necessary non-linearity in the model. By introducing non-linearity, tanh allows the network to learn complex patterns in the data, which is crucial for tasks involving sequence and pattern recognition.

Output Calculation

$$\hat{y} = \lambda(V h_t)$$

- V is a weight matrix that transforms the hidden state h_t to the output space.
- λ represents some function, possibly a transformation or activation function like softmax, which adapts the raw output of the network to the specific needs of the task (e.g., classification, regression).

12.2.2 Forward Propagation

The network processes inputs through time.
For each time step t, the hidden state h_t is updated based on the current input x_t and the previous hidden state h_{t-1} using the transition weights U and W, respectively.
The output $\hat{y}_t$ at each time step is then computed using the output weights V.

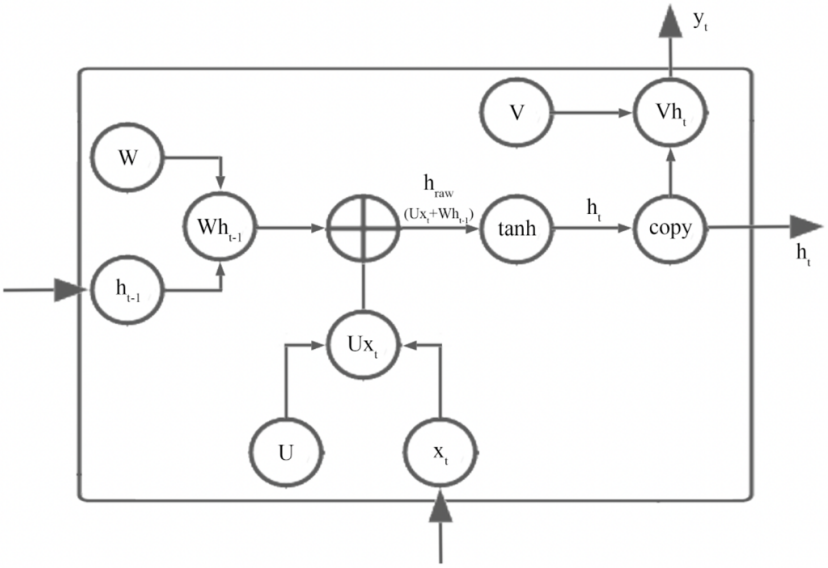


Figure 3: RNN Architecture Design

What are the dimensions of the weight matrices?

We define:

- n_{inputs}: Number of input features (the number of features/variables in the dataset to predict y_t)
- $n_{neurons}$: Number of hidden units (this is arbitrarily chosen based on the model's complexity for the project)
- $n_{outputs}$: Number of output units (the output we are deriving)

For stock price of a one company:
n_{inputs} = 1 (only one stock price list we are looking at)
$n_{outputs}$ = number of forecasting units in the future

Input vector X_t: $n_{inputs} \times 1$
Hidden state vector h_t: $n_{neurons} \times 1$.
Weight matrices:

- U: $n_{neurons} \times n_{inputs}$.
- W: $n_{neurons} \times n_{neurons}$.
- V: $n_{outputs} \times n_{neurons}$.

The function λ acts on the vector Vh_t, where V is the weight matrix that transforms the hidden state into the output space. The choice of λ is crucial because it shapes the final form of the output depending on the specific requirements of the task at hand.

λ can be any suitable function, including nonlinear activation functions, linear transformations, or even another neural network model. This flexibility allows the RNN to be tailored to a wide range of applications, from simple regression tasks to complex classification problems.

- Sigmoid: binary probability distribution
- Softmax: categorical probability distribution
- ReLU: positive real-value output
- Identity function: real-value output

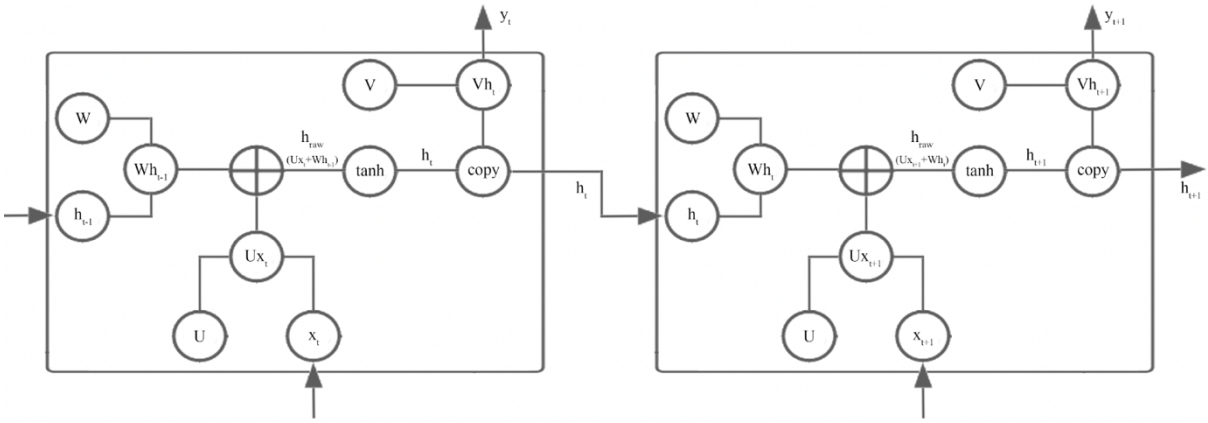
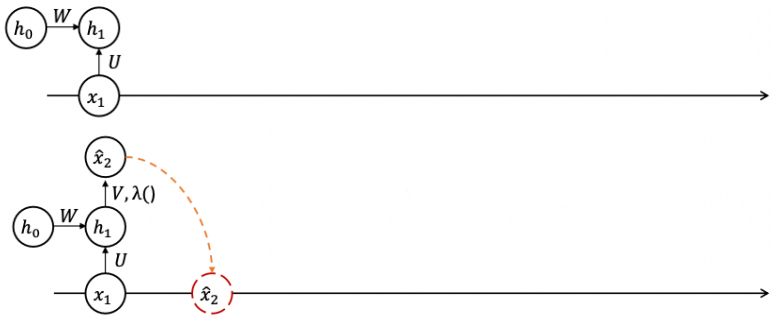
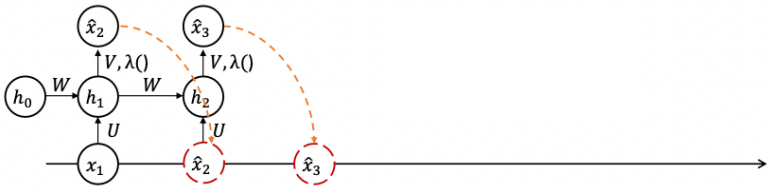


Figure 4: RNN Architecture Linked Design



$$h_1 = \tanh(Ux_1 + Wh_1)$$

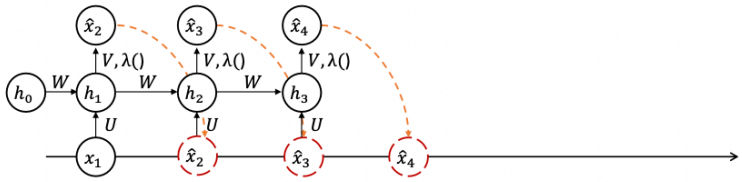
$$\hat{x}_2 = \hat{y} = \lambda(Vh_2)$$



In prediction for multiple steps ahead, predicted value $\hat{x}_2$ from previous step is considered as input x_2 at time step 2.

$$h_2 = \tanh(U\hat{x}_2 + Wh_1)$$

$$\hat{x}_3 = \hat{y} = \lambda(Vh_2)$$



$$h_3 = \tanh(U\hat{x}_3 + Wh_2)$$

$$\hat{x}_4 = \hat{y} = \lambda(Vh_3)$$

How do you train an RNN?

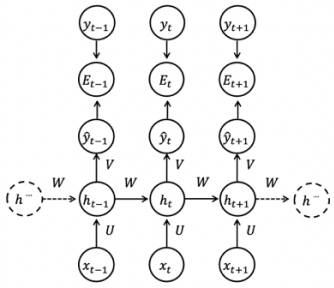


Figure 5: RNN Model

- True Target (y^t): This is the actual output that the model is supposed to predict, based on the input data. It's what the training data says should happen at each time step.
- Predicted Output ($\hat{y}^t$): This is the output that the RNN predicts based on the input it has received. The prediction is derived from the final transformation function λ applied to the hidden states, which typically adjusts the raw output to suit the problem's specific needs (like classification or regression).
- Error (Loss) E_t: This represents the discrepancy between the true target y^t and the predicted output $\hat{y}$. The goal of training is to minimise this error across all predictions.
    - Binary Classification: Binary Cross Entropy
    - Categorical Classification: Cross Entropy
    - Regression: Mean Squared Error

12.2.3 Back Propagation Proofs

Back Propagation Through Time (BPTT)

BPTT is an adaptation of the traditional backpropagation algorithm used for training feedforward neural networks. It's tailored to handle the sequential nature of RNNs.

How does it work?

Error Calculation:

At each time step, the output $\hat{y}_t$ is compared to the true target y_t, and an error $\mathbf{E_t}$ is computed.
This error takes the predicted and the actual as inputs and is typically calculated using a loss function appropriate for the task (e.g., mean squared error for regression).

Backward Pass:

Learning Parameters for RNN

The goal of an RNN is to find:

$$\frac{\partial E_t}{\partial W_t}$$

$$\frac{\partial E}{\partial W} = \frac{\partial E_{t+1}}{\partial W} + \frac{\partial E_{t+2}}{\partial W} \dots = \sum_t \frac{\partial E_t}{\partial W}$$

Where t+1, t+2, … t+k, where k is the number of values in the future.

This term represents the partial derivative of the error with respect to the weight matrix. It measures how a small change in the weights affects the error term. In other words, this term indicates how the weights should be adjusted to minimise the error.

We use this gradient information from $\frac{\partial E_t}{\partial W_t}$ to update W via optimisation algorithm SGD.
This is based on the formula below.

$$W_{new} = W_{old} - \eta \frac{\partial E_t}{\partial W}$$

But how does this SGD work exactly? How does it improve W?

$\frac{\partial E_t}{\partial W}$ is a numerical value.

- If the gradient is positive, it indicates W increases with E_t. If it is negative, increasing W, decreases E_t.
- The size of the gradient tells how sensitive E_t is to changes in W.

η is the learning rate, a small positive number that controls how much we adjust W by.
By subtracting $\eta \frac{\partial E_t}{\partial W}$, you are moving W in the direction that most reduces E_t because you're moving against the gradient.
This update isn't a one-time calculation but part of an iterative process. With each iteration (or epoch in training terminology), W is updated to gradually reduce E_t. Over many iterations, this process is expected to converge to a set of weights W that minimise E_t, although only reaching a local minimum is guaranteed.

To derive this formula, we can further break it down to the following:

$$\frac{\partial E_t}{\partial W_t} = \frac{\partial E_t}{\partial h_k} \cdot \frac{\partial h_k}{\partial W_t}$$

Remember, the hidden layer is where most of the computations happen to determine relationships and patterns. Thus, we use this to determine changes in the error.
Here, $\frac{\partial E_t}{\partial h_k}$ provides the gradient of the error as influenced by the hidden states and $\frac{\partial h_k}{\partial W_t}$ adjusts this influence based on how the weights affect those hidden states.

These are the gradients derived which are necessary for training a RNN through the backpropagation through time (BPTT) method.

Please reference the following multiplication types:

- $\cdot$: to denote dot product
- $*$: to denote element wise multiplication
- No symbol implies normal multiplication

We follow the back propagation arrows to calculate the respective gradients.

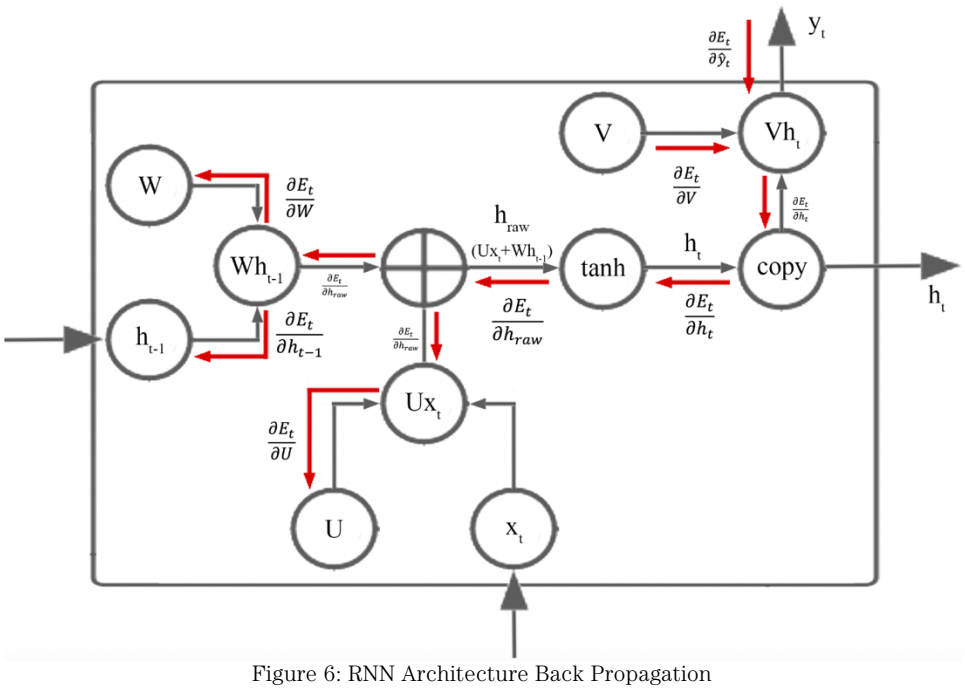


Figure 6: RNN Architecture Back Propagation

We are finding $\frac{\partial E_t}{\partial \hat{y}_t}$.

$$E = E\left(\hat{y}_1^{(\tau)}, \ldots, \hat{y}_L^{(\tau)}\right) = E\left(\hat{y}_1^{(\tau)}\left(h_i^{(\tau)}\right), \ldots, \hat{y}_L^{(\tau)}\left(h_i^{(\tau)}\right)\right)$$

Why derive with respect to $\hat{y}_t$?

- The actual target y_t is a fixed value and does not change during training. Hence, derivatives with respect to y_t do not make sense in the context of adjusting network weights. We are interested in how adjustable parameters (like h_t W, and U) influence errors, not fixed data points.
- The gradient $\frac{\partial E_t}{\partial h_t}$ as derived connects how changes in the hidden states via h_t (influenced by W and U) propagate to influence the prediction ($\hat{y}_t$), which directly impacts the error.

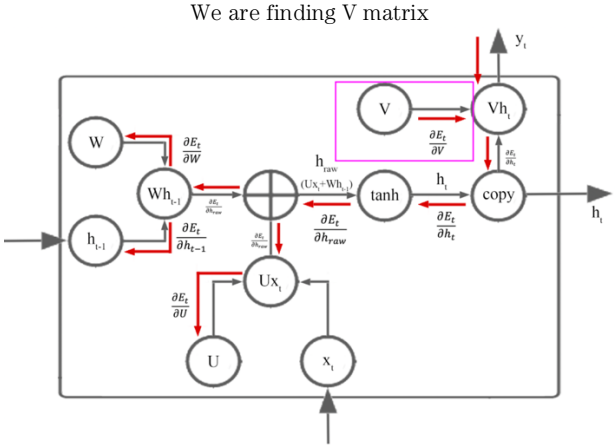
For simplicity sake of this proof, we assume E_t is simple squared error.

$$E_t = \frac{1}{2}(\hat{y}_t - y_t)^2 = \frac{1}{2}[(\hat{y}_t - y_t)(\hat{y}_t - y_t)] = \frac{1}{2}[\hat{y}_t^2 - 2\hat{y}_t y_t + y_t^2] = \frac{1}{2}\hat{y}_t^2 - y_t\hat{y}_t + \frac{1}{2}y_t^2$$

From this we can derive

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

We are finding V matrix



Remember we follow the upstream/downstream method.

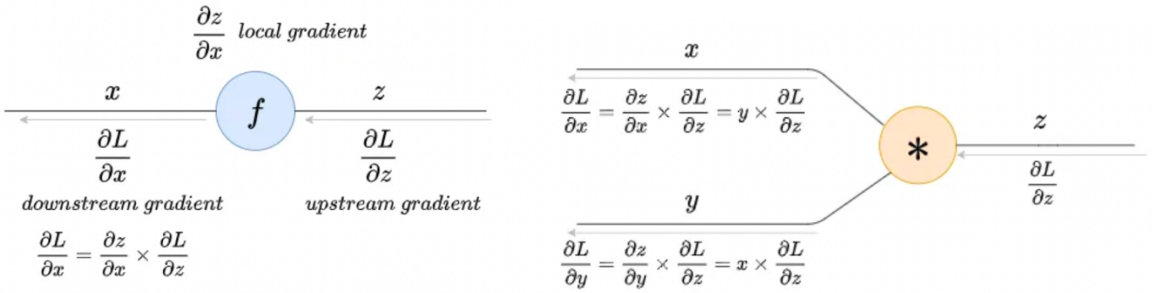


Figure 7: Downstream / Upstream Calculation Graph (Krishna, N., 2022)

$$downstream\ gradient = local\ gradient \times upstream\ gradient$$

Therefore, because the downward arrow from $\hat{y}_t$ is approaching V, we utilise this.

$$\frac{\partial E_t}{\partial V_t} = \frac{\partial \hat{y}_t}{\partial V_t} \cdot \frac{\partial E_t}{\partial \hat{y}_t}$$

Using basic differentiation,

$$\frac{\partial \hat{y}_t}{\partial V} = h_t V = h_t$$

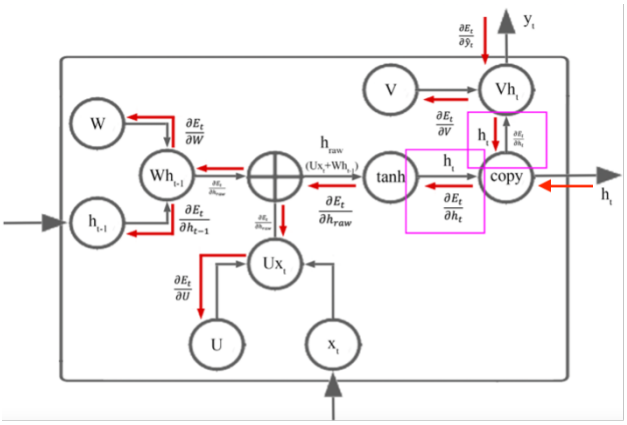
$$\frac{\partial E_t}{\partial V_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot h_t$$

Remember, we derived $\frac{\partial E_t}{\partial \hat{y}_t}$,

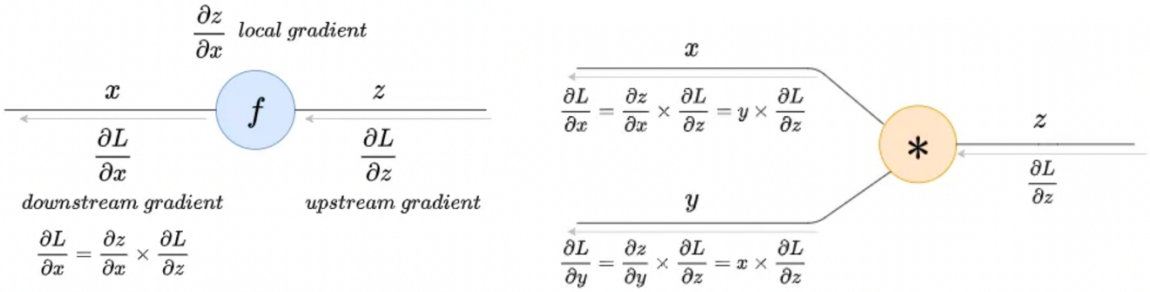
$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

$$\frac{\partial E_t}{\partial V_t} = (\hat{y}_t - y_t) \cdot {h_t}^T$$

We are finding h_t



Remember we follow the upstream/downstream method.



$$downstream\ gradient = local\ gradient \times upstream\ gradient$$

Therefore, because the downward arrow from $\hat{y}_t$ is approaching h_t (which is also = to $copy$), we utilise this.

$$\frac{\partial E_t}{\partial h_t} = \left(\frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t}\right) + h_{next}$$

Where (for simplicity)

$$h_{next} = (\frac{\partial E_t}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t})$$

Or more accurately put (take into account all the h values ahead):

$$h_{next} = \left(\frac{\partial E_t}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t}\right) + \left(\frac{\partial E_t}{\partial h_{t+2}} \cdot \frac{\partial h_{t+2}}{\partial h_t}\right) + \cdots = \sum_{m=1}^{\infty} \left(\frac{\partial E_t}{\partial h_{t+m}} \cdot \frac{\partial h_{t+m}}{\partial h_t}\right)$$

Remember, we derived $\frac{\partial E_t}{\partial \hat{y}_t}$,

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

Remember,

$$\hat{y}_t = \lambda(Vh_t)$$

Important Note: For calculation simplicity, we assume the transformation from h_t to $\hat{y}_t$ performs a linear transformation. But it is important to note that λ could be any suitable function such as softmax for classification tasks or identity for regression, etc.

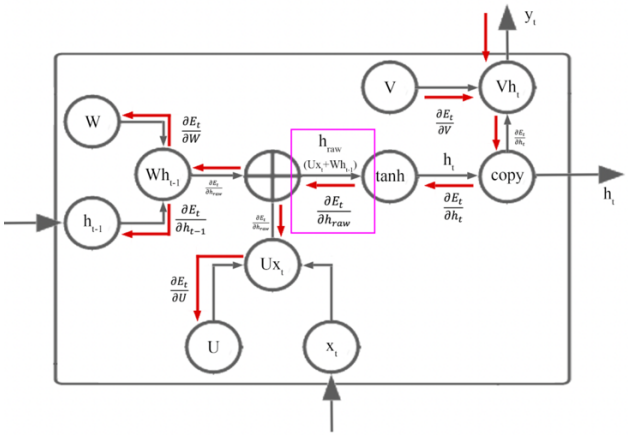
$$\hat{y}_t = Vh_t$$

$$\frac{\partial \hat{y}_t}{\partial h_t} = V$$

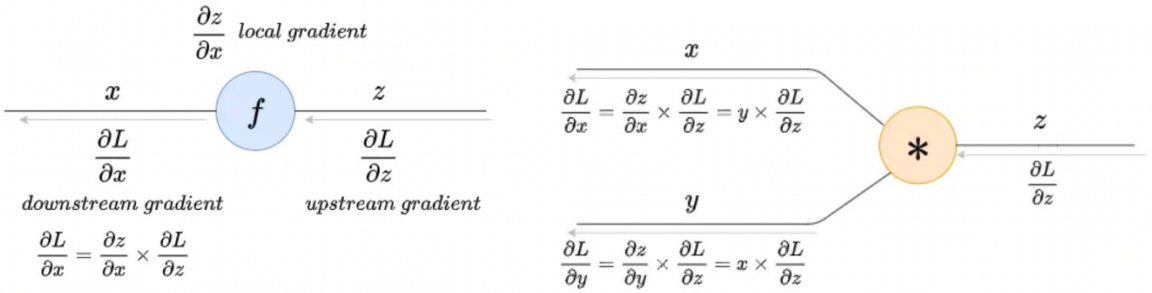
Therefore,

$$\beta_k = \frac{\partial E_t}{\partial h_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} + h_{next} = (\hat{y}_t - y_t)V + h_{next}$$

We are finding h_{raw}



Remember we follow the upstream/downstream method.



$$downstream\ gradient = local\ gradient \times upstream\ gradient$$

$$\frac{\partial E_t}{\partial h_{raw}} = \frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}$$

Therefore, because the downward arrow from h_t is approaching h_{raw}, we utilise this.

Let's first substitute the following derivative (that we found above) into the equation.

$$\frac{\partial E_t}{\partial h_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t}$$

$$\frac{\partial E_t}{\partial h_{raw}} = \frac{\partial h_t}{\partial h_{raw}} \cdot \left(\frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t}\right)$$

Remember that,

$$h_t = \tanh(h_{raw})$$

Therefore,

$$\frac{\partial h_t}{\partial h_{raw}} = 1 - (\tanh(h_{raw}))^2 = 1 - {h_t}^2$$

And remember we have:

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

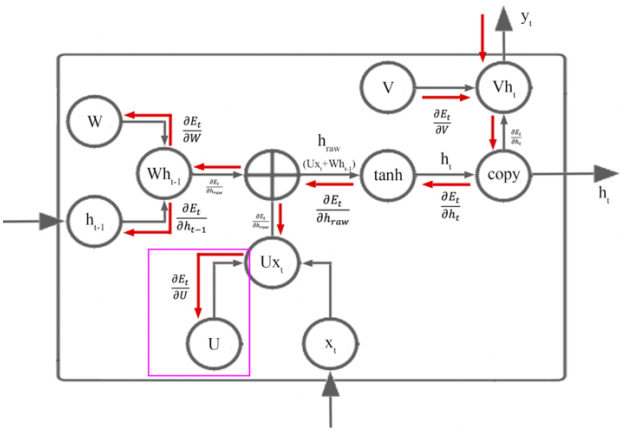
$$\frac{\partial \hat{y}_t}{\partial h_t} = V$$

Finally, we combine the two:

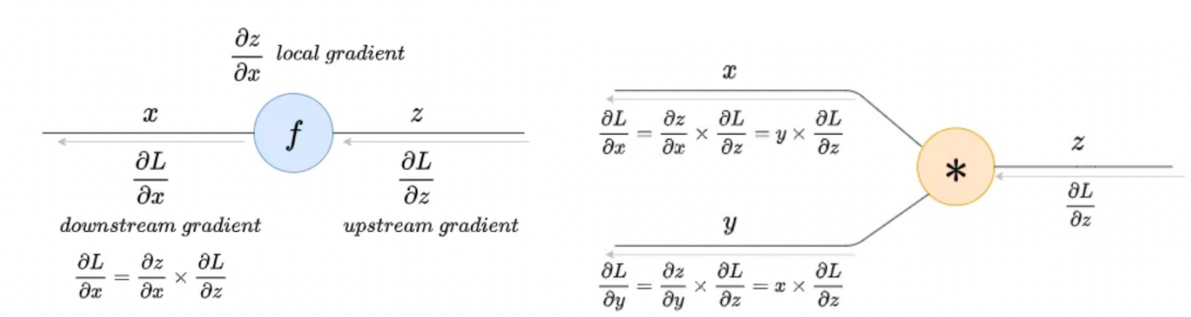
$$\frac{\partial E_t}{\partial h_{raw}} = \frac{\partial h_t}{\partial h_{raw}} \cdot \left(\frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t}\right)$$

$$\frac{\partial E_t}{\partial h_{raw}} = \left[1 - {h_t}^2\right] * [(\hat{y}_t - y_t)V]$$

Calculate for U matrix



Remember we follow the upstream/downstream method.



$$downstream\ gradient = local\ gradient \times upstream\ gradient$$

Therefore, because the downward arrow from h_{raw} is approaching U, we utilise this.

$$\frac{\partial E_t}{\partial U} = \frac{\partial h_{raw}}{\partial U} \cdot \frac{\partial E_t}{\partial h_{raw}}$$

$$h_{raw} = Ux_t + Wh_{t-1}$$

We derive h_{raw} with respect to U

$$\frac{\partial h_{raw}}{\partial U} = x_t$$

Remember we have the following terms derived above:

$$\frac{\partial E_t}{\partial h_{raw}} = \frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}$$

Therefore, we have:

$$\frac{\partial E_t}{\partial U} = \frac{\partial h_{raw}}{\partial U} \cdot \left(\frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}\right)$$

And substituting the following

$$\frac{\partial h_t}{\partial h_{raw}} = 1 - (\tanh(h_{raw}))^2 = 1 - {h_t}^2$$

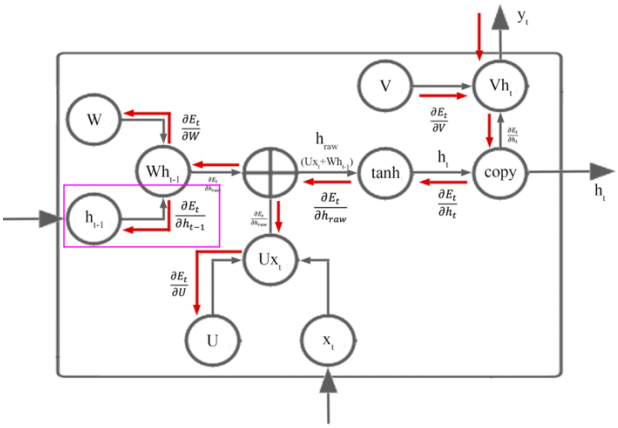
$$\frac{\partial E}{\partial h_t} = (\hat{y}_t - y_t)V$$

$$\frac{\partial h_{raw}}{\partial U} = x_t$$

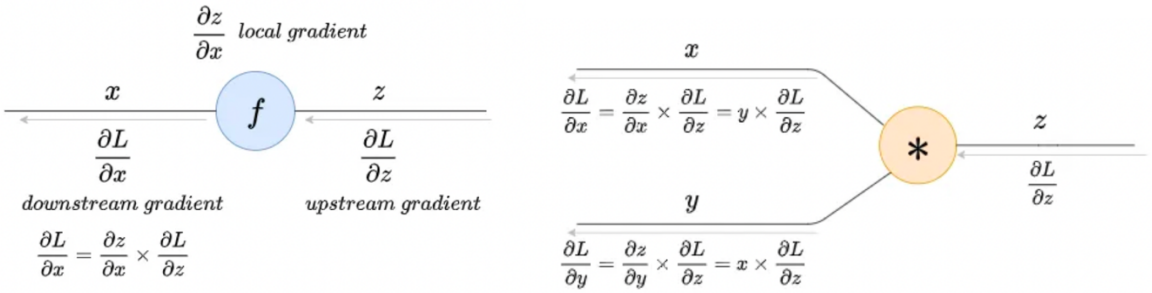
We get:

$$\frac{\partial E_t}{\partial U} = \left[1 - {h_t}^2\right] \cdot [(\hat{y}_t - y_t)V] \cdot [{x_t}^T]$$

We are finding h_{t-1}



Remember we follow the upstream/downstream method.



$$downstream\ gradient = local\ gradient \times upstream\ gradient$$

Therefore, because the downward arrow from h_{raw} is approaching h_{t-1}, we utilise this.

$$\frac{\partial E_t}{\partial h_{t-1}} = \frac{\partial h_{raw}}{\partial h_{t-1}} \cdot \frac{\partial E}{\partial h_{raw}}$$

Remember,

$$h_{raw} = Ux_t + Wh_{t-1}$$

We derive h_{raw} with respect to h_{t-1}

$$\frac{\partial h_{raw}}{\partial h_{t-1}} = W$$

Remember we have the following terms derived above:

$$\frac{\partial E_t}{\partial h_{raw}} = \frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}$$

Therefore, we have:

$$\frac{\partial E_t}{\partial h_{t-1}} = \frac{\partial h_{raw}}{\partial h_{t-1}} \cdot \left(\frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}\right)$$

And substituting the following

$$\frac{\partial h_t}{\partial h_{raw}} = 1 - (\tanh(h_{raw}))^2 = 1 - {h_t}^2$$

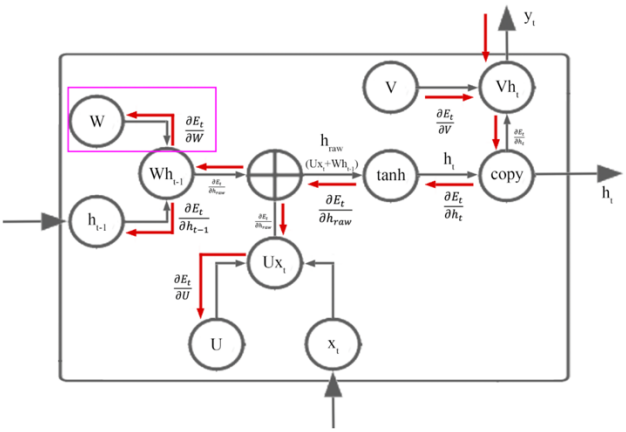
$$\frac{\partial E}{\partial h_t} = (\hat{y}_t - y_t)V$$

$$\frac{\partial h_{raw}}{\partial h_{t-1}} = W$$

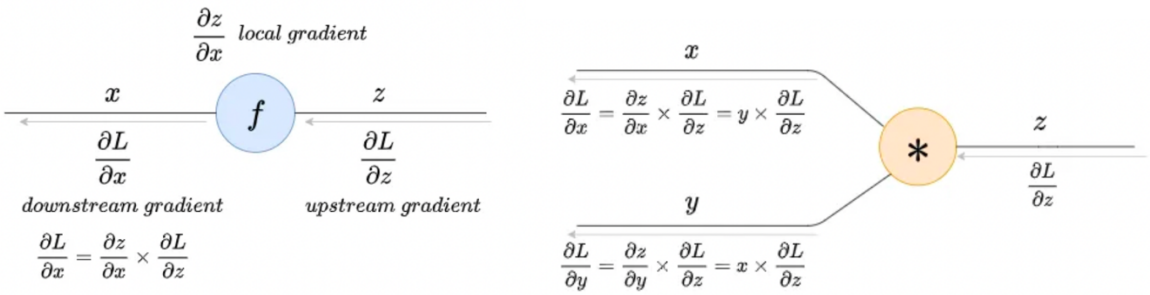
We get:

$$\frac{\partial E_t}{\partial U} = \left[1 - {h_t}^2\right] \cdot [(\hat{y}_t - y_t)V] \cdot [W]$$

Calculate for W matrix



Remember we follow the upstream/downstream method.



$$downstream\ gradient = local\ gradient \times upstream\ gradient$$

Therefore, because the downward arrow from h_{raw} is approaching W, we utilise this.

$$\frac{\partial E_t}{\partial W} = \frac{\partial h_{raw}}{\partial W} \cdot \frac{\partial E_t}{\partial h_{raw}}$$

$$h_{raw} = Ux_t + Wh_{t-1}$$

We derive h_{raw} with respect to U

$$\frac{\partial h_{raw}}{\partial W} = h_{t-1}$$

Remember we have the following terms derived above:

$$\frac{\partial E_t}{\partial h_{raw}} = \frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}$$

Therefore, we have:

$$\frac{\partial E_t}{\partial W} = \frac{\partial h_{raw}}{\partial W} \cdot \left(\frac{\partial h_t}{\partial h_{raw}} \cdot \frac{\partial E}{\partial h_t}\right)$$

And substituting the following

$$\frac{\partial h_t}{\partial h_{raw}} = 1 - (\tanh(h_{raw}))^2 = 1 - {h_t}^2$$

$$\frac{\partial E}{\partial h_t} = (\hat{y}_t - y_t)V$$

$$\frac{\partial h_{raw}}{\partial W} = h_{t-1}$$

We get:

$$\frac{\partial E_t}{\partial W} = \left[1 - {h_t}^2\right] \cdot [(\hat{y}_t - y_t)V] \cdot \left[{h_{t-1}}^T\right]$$

12.2.4 Vanishing Gradient & Solution

Vanishing and Exploding Gradients are significant issues that occur during the training of Recurrent Neural Networks (RNNs), particularly with long sequences. They affect the training stability and efficiency, making it difficult for the network to learn long-term dependencies.

Gradient Calculation through Time:
In RNNs, the gradient of the loss with respect to the parameters (i.e weight matrices) is computed through the chain rule applied over multiple time steps.
The gradient at each time step is influenced by the gradient at the previous time steps, leading to a product of gradients (Jacobian matrices).

We start at time t. We want to forecast t+1, t+2, t+3, t+4. We are forecasting 4 values in the future.

The partial derivative tells us how the current hidden state (h_{t+1}) is to changes in the previous hidden state (h_t).
The Jacobian matrix describes how small changes in the hidden state at time step j affect the hidden state at time step $j+1$.

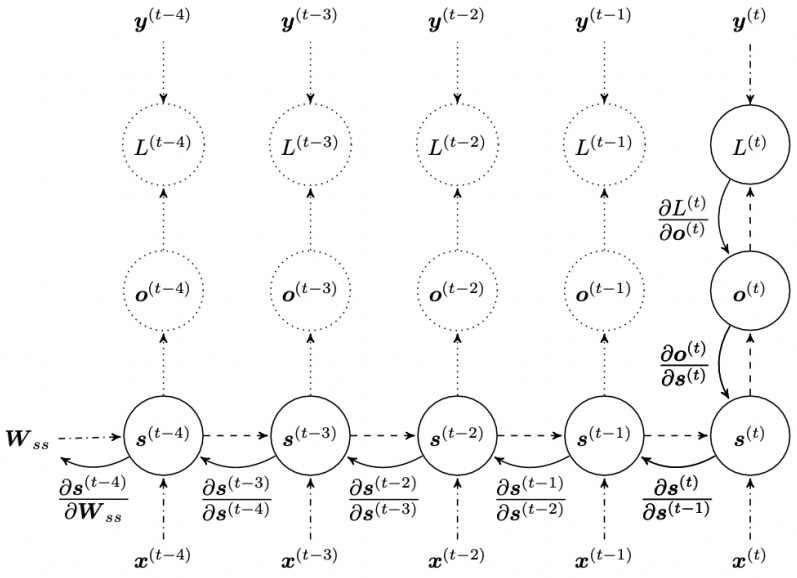


Figure 8: Partial derivative chain for a basic RNN for a history of 4 time steps (Borne, K., 2016)

We will be doing derivation for W, but the same mathematical proof can be applied for weight matrices U and V (as they are also carried over).

We sum the error at each time-step. That is $\frac{\partial E_t}{\partial W}$ for every time step, t, is computed and accumulated.

$$\frac{\partial E}{\partial W} = \frac{\partial E_{t+1}}{\partial W} + \frac{\partial E_{t+2}}{\partial W} \dots = \sum_{t}^{T} \frac{\partial E_t}{\partial W}$$

Remember we have the following derivation:

$$\frac{\partial E_t}{\partial W} = \frac{\partial h_{(raw,t)}}{\partial W} \cdot \frac{\partial E_t}{\partial h_{(raw,t)}}$$

We substitute this into the summation equation to get:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial h_{(raw,t)}}{\partial W} \cdot \frac{\partial E_T}{\partial h_{(raw,t)}}$$

For simplicity in presenting, we will assume we have max 3 RNN units ($T=3$) in the model.
Thus, the $\frac{\partial E_T}{\partial h_{(raw,t)}}$ in the summation will be respectively:

$$\frac{\partial E_3}{\partial h_{(raw,1)}} = \frac{\partial h_{(raw,2)}}{\partial h_{(raw,1)}} \cdot \frac{\partial h_{(raw,3)}}{\partial h_{(raw,2)}} \cdot \frac{\partial E_3}{\partial h_{(raw,3)}}$$

$$\frac{\partial E_3}{\partial h_{(raw,2)}} = \frac{\partial h_{(raw,3)}}{\partial h_{(raw,2)}} \cdot \frac{\partial E_3}{\partial h_{(raw,3)}}$$

$$\frac{\partial E_3}{\partial h_{(raw,3)}} = \frac{\partial E_3}{\partial \hat{y}_3} \cdot \frac{\partial \hat{y}_3}{\partial h_{(raw,3)}}$$

Important note: We can do $\frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}}$ because $h_{(raw,k)}$ is in reference/is directly linked with the previous step, $h_{(raw,k-1)}$. Thus, we can apply this with the chain rule to derive the relationship and link with $\frac{\partial E_t}{\partial W}$.

Therefore, we can rewrite in this format (remember summation, T is inclusive).
It is important to note that once t=T, we simply multiply by: $\frac{\partial E_T}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t}$ (as per the small T=3 example above).

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial h_{(raw,t)}}{\partial W} \cdot \prod_{k=t+1}^{T} \frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}} \cdot \frac{\partial E_T}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial h_T}$$

When T=3, then we have (for reference):

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{3} \frac{\partial h_{(raw,t)}}{\partial W} \cdot \prod_{k=t+1}^{T} \frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}} \cdot \frac{\partial E_3}{\partial \hat{y}_3} \cdot \frac{\partial \hat{y}_3}{\partial h_3}$$

For better visual and understanding, we can define:

$$\frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}} = \prod_{k=t+1}^{T} \frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}}$$

And also substituting

$$\frac{\partial E_T}{\partial h_{(raw,T)}} = \frac{\partial E_T}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial h_{(raw,T)}}$$

We turn,

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial h_{(raw,t)}}{\partial W} \cdot \frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}} \cdot \frac{\partial E_T}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial h_{(raw,T)}}$$

Into:

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial h_{(raw,t)}}{\partial W} \cdot \frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}} \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$$

Remember the following equations:

$$h_{(raw,t)} = Ux_t + Wh_{t-1}$$

$$h_{(raw,t-1)} = Ux_{t-1} + Wh_{t-2}$$

$$\frac{\partial h_{(raw,t)}}{\partial h_{(raw,t-1)}} = W$$

Sometimes we can derive the proof with h_t which give:

$$h_t = \tanh(Ux_t + Wh_{t-1})$$

$$h_{t-1} = \tanh(Ux_{t-1} + Wh_{t-2})$$

We are applying the chain rule here:

$$\frac{\partial h_t}{\partial h_{t-1}} = [1 - (\tanh(Ux_t + Wh_{t-1}))^2] \cdot W$$

Remember the equation we derived above:

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial h_{(raw,t)}}{\partial W} \cdot \frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}} \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$$

The expressions $\frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}}$ and $\frac{\partial E_T}{\partial h_{(raw,T)}}$ can be multiplied together through chain rule to get $\frac{\partial E_T}{\partial h_{(raw,t)}}$.

$$\frac{\partial E_T}{\partial h_{(raw,t)}} = \frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}} \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$$

We also derived this expression above.

$$\frac{\partial h_{(raw,T)}}{\partial h_{(raw,t)}} = \prod_{k=t+1}^{T} \frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}}$$

We substitute this into the expression above.

$$\frac{\partial E_T}{\partial h_{(raw,t)}} = \prod_{k=t+1}^{T} \frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}} \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$$

As per our chain rule derivation above, we substitute how $\frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}} = W$ (assume h_{raw} for simplicity but can also substitute h_t).

$$\frac{\partial E_T}{\partial h_{(raw,t)}} = \prod_{k=t+1}^{T} W \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$$

Now given this expression: $\prod_{k=t+1}^{T} W \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$, **if W is small, then this term gets vanishingly small as we repeatingly multiply by itself T times.**

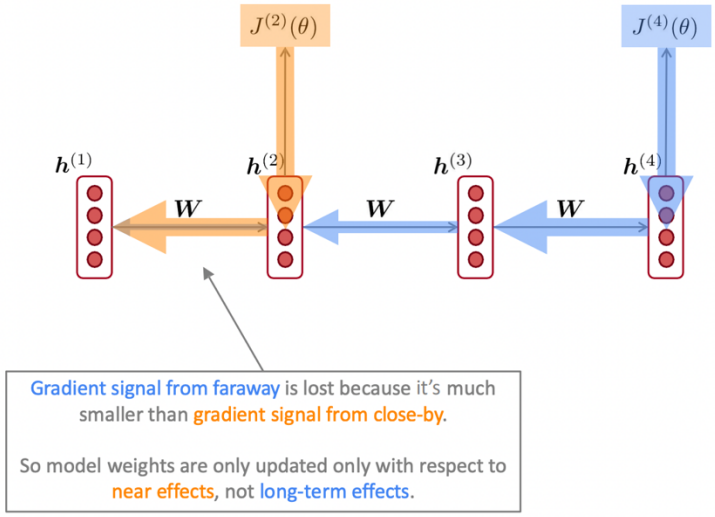


Figure 9: Vanishing Gradient Intuition Diagram (Kovashka, A., 2021)

The Jacobian matrix in the context of RNNs represents how the hidden state at one timestep is affected by the hidden state at the previous timestep. More formally, if you have a hidden state $h_{(raw,t)}$ at time t, which is computed based on the previous state $h_{(raw,t-1)}$, the Jacobian matrix $\frac{\partial h_{(raw,t)}}{\partial h_{(raw,t-1)}}$ tells you how changes in $h_{(raw,t-1)}$ influence changes in $h_{(raw,t)}$.

Eigenvalues ($\boldsymbol{\lambda}$) of the Jacobian matrix describe the rate of expansion or contraction in different directions in the state space.

- If an eigenvalue is greater than 1, it indicates that variations along the corresponding eigenvector grow exponentially as they propagate through timesteps, which can lead to exploding gradients.
- If an eigenvalue is less than 1, it indicates decay in those variations, potentially leading to vanishing gradients.

Eigenvectors ($\boldsymbol{q}$) of the Jacobian matrix correspond to the directions in the state space along which these expansions or contractions occur. Each eigenvector provides a "principal direction" of change.

$$\prod_{k=t+1}^{T} \frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}}$$

This product can exponentially amplify or diminish gradients depending on the magnitudes of the Jacobian's eigenvalues.

We can write $\frac{\partial E_T}{\partial h_{(raw,t)}}$ which is $\prod_{k=t+1}^{T} W \cdot \frac{\partial E_T}{\partial h_{(raw,T)}}$ using eigenvectors of W.

If performing eigen decomposition on each Jacobian matrix $\frac{\partial h_{(raw,k)}}{\partial h_{(raw,k-1)}}$, you will get eigenvalues $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_n$.

Eigen decomposition is a form of matrix decomposition where a matrix $\boldsymbol{J}$ is broken down into the product of its eigenvectors and eigenvalues.

If $\boldsymbol{W}$ is the Jacobian matrix, which describes how the hidden state at one timestep is derived from the state at the previous timestep, eigen decomposition involves finding a matrix $\boldsymbol{V}$ (whose columns are the eigenvectors of ($\boldsymbol{W}$) and a diagonal matrix $\boldsymbol{\Lambda}$ (whose diagonal elements are the eigenvalues of ($\boldsymbol{W}$) such that:

$$W = V\boldsymbol{\Lambda}\mathbf{V^{-1}}$$
$$\boldsymbol{\Lambda} = \mathbf{diag}(\boldsymbol{\lambda})$$

$$W^t = (V \cdot \mathbf{diag}(\boldsymbol{\lambda}) \cdot \mathbf{V^{-1}})^t = V \cdot \mathbf{diag}(\boldsymbol{\lambda^t}) \cdot \mathbf{V^{-1}}$$

Any eigenvalues $\boldsymbol{\lambda_i}$ that are not near an absolute value of 1 will either:

- explode if they are greater than 1 in magnitude
- vanish if they are less than 1 in magnitude

12.3 LSTM (Long Short Term Memory)

12.3.1 Background

Long Short-Term Memory networks, commonly known as LSTMs, are a special type of Recurrent Neural Network (RNN) that excel at learning from sequences of data. Unlike traditional RNNs, LSTMs are designed to remember information over long periods, making them particularly effective for tasks involving time-series data, natural language processing, and any application where context over time is crucial.

Basic RNNs suffer from the vanishing gradient problem, which makes it difficult for them to learn long-term dependencies in data. They tend to forget information quickly, retaining only recent inputs. LSTMs address this limitation by introducing a more complex cell structure that can maintain information over extended time intervals.

12.3.2 Formulas

An LSTM cell consists of several components that work together to regulate the flow of information:

- Forget Gate (f_t)
- Input Gate (i_t)
- Cell State (C_t)
- Output Gate (o_t)

These gates use activation functions like the sigmoid (σ) and hyperbolic tangent ($tanh$) to control which information is forgotten, updated, or passed on to the next time step.

Common notations used to understand LSTMs:

- h_t: Hidden state at time t
- h_{t-1}: Hidden state at time $t-1$
- x_t: Input at time t
- C_t: Cell state at time t
- W and b: Weights and biases for different gates
- σ: Sigmoid activation function
- tanh: Hyperbolic tangent activation function
- $\odot$: Element-wise (Hadamard) multiplication

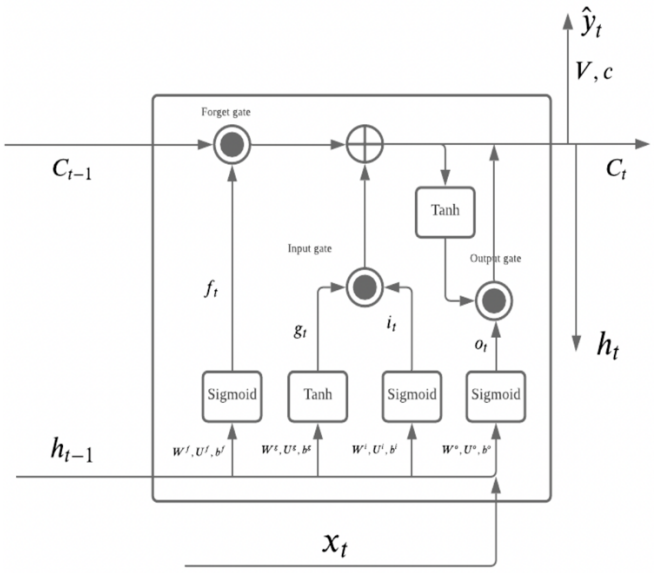
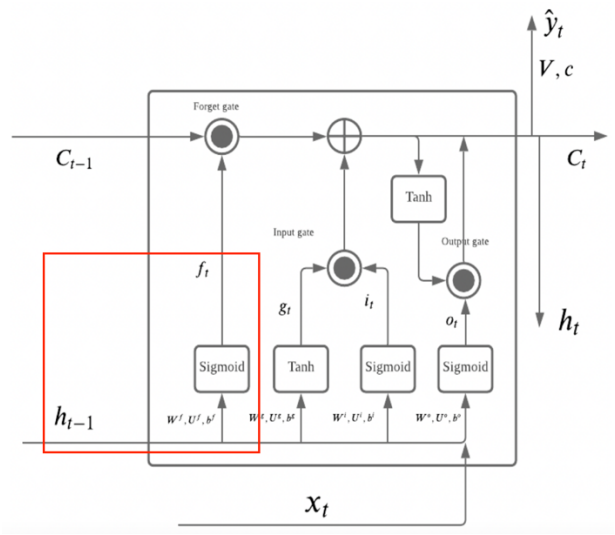


Figure 10: LSTM Architecture (Fang et al., 2021)

Forget Gate (f_t)



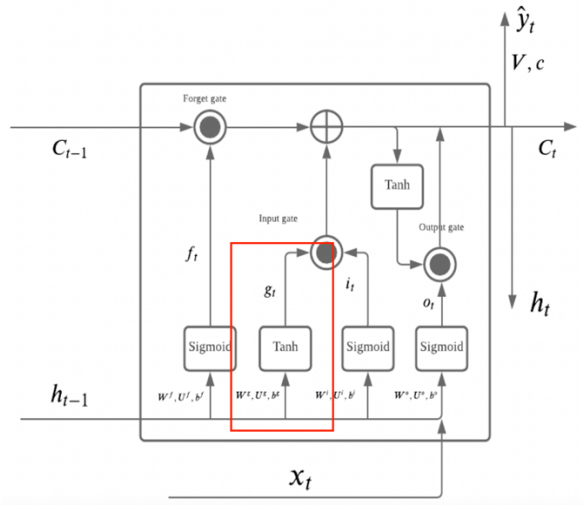
Purpose: Decides what information to discard from the cell state (C_{t-1})

$$f_t = \sigma(U_f x_t \cdot W_f h_{t-1} + b_f)$$

Remember: $sigmoid = \sigma$

Input: Concatenation of h_{t-1} and x_t.
Process: Applies a sigmoid activation function to produce values between 0 and 1.
Output: A vector f_t that determines which pieces of the cell state C_{t-1} forget (values close to 0) or retain (values close to 1).

Candidate Values (g_t)

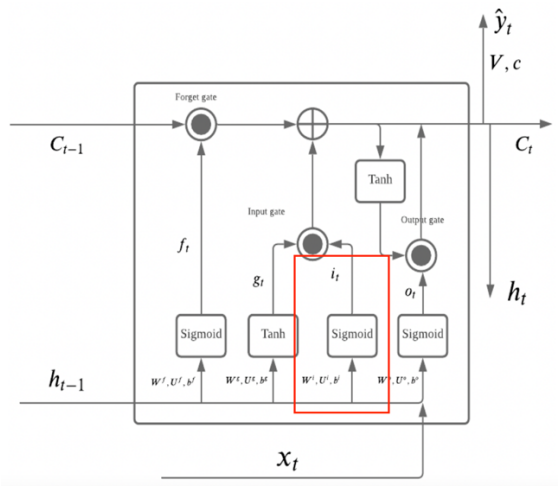


Purpose: Decides what new information to store in the cell state.

$$g_t = \tanh\left(U_g x_t + W_g h_{t-1} + b_g\right)$$

Candidate Values (g_t): Represents potential new information to be added to the cell state, generated by passing the concatenated input through a $tanh$ activation to produce values between -1 and 1.

Input Gate (i_t)



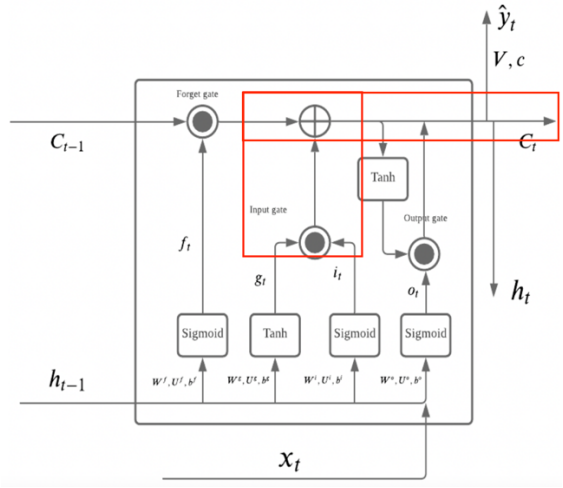
Purpose: Controls how much g_t be remembered

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

Remember: $sigmoid = \sigma$

Input Gate (i_t): Acts as a filter to determine which values from g_t will be used to update the cell state.

Updating the C_t



Purpose: Maintains the long-term memory of the network.

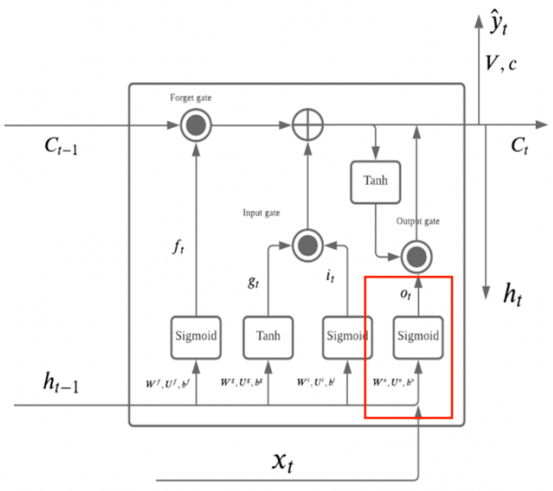
$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$

First Term ($f_t \odot C_{t-1}$): Applies the forget gate to the previous cell state, effectively discarding or retaining information.

Second Term ($i_t \odot g_t$): Adds new, relevant information to the cell state based on the input gate and candidate values.

Result: An updated cell state C_t that carries forward important information.

Output Gate (o_t)

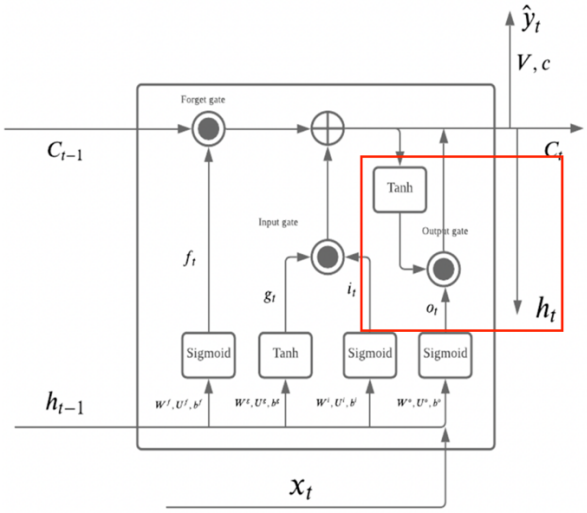


Purpose: Determines what part of the cell state (C_t) to output and passes it to the next time step and layers.

$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o)$$

Output Gate (o_t): Decides which parts of the cell state to output.

Hidden State (h_t)

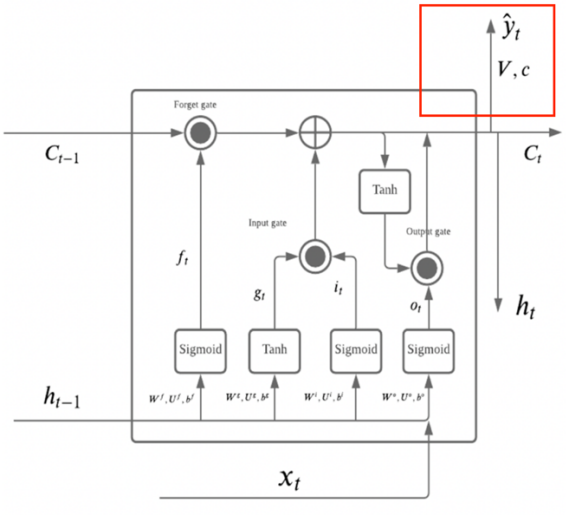


Purpose: Determines what part of the cell state to output and passes it to the next time step and layers.

$$h_t = o_t \odot \tanh(C_t)$$

Hidden State (h_t): The final output at time t, combining the cell state (passed through $tanh$) and the output gate.

Generating Final Output ($\hat{y}_t$)



Purpose: Produces the prediction or output for the current time step.

For regression:

$$\hat{y}_t = h_t \cdot V + c$$

For classification:

$$\hat{y}_t = \text{softmax}(h_t \cdot V + c)$$

12.3.2 Back Propagation Proofs

Note: ∂h_{next} and ∂C_{next} next represent the sum of gradients from future time steps.

We are finding $\frac{\partial E_t}{\partial \hat{y}_t}$

$$E = E\left(\hat{y}_1^{(\tau)}, \dots, \hat{y}_L^{(\tau)}\right) = E\left(\hat{y}_1^{(\tau)}\left(h_i^{(\tau)}\right), \dots, \hat{y}_L^{(\tau)}\left(h_i^{(\tau)}\right)\right)$$

Why derive with respect to $\hat{y}_t$?

- The actual target y_t is a fixed value and does not change during training. Hence, derivatives with respect to y_t do not make sense in the context of adjusting network weights. We are interested in how adjustable parameters (like h_t W, and U) influence errors, not fixed data points.
- The gradient $\frac{\partial E_t}{\partial h_t}$ as derived connects how changes in the hidden states via h_t (influenced by W and U) propagate to influence the prediction ($\hat{y}_t$), which directly impacts the error.

For simplicity sake of this proof, we assume E_t is simple squared error.

$$E_t = \frac{1}{2}(\hat{y}_t - y_t)^2 = \frac{1}{2}[(\hat{y}_t - y_t)(\hat{y}_t - y_t)] = \frac{1}{2}[\hat{y}_t^2 - 2\hat{y}_t y_t + y_t^2] = \frac{1}{2}\hat{y}_t^2 - y_t\hat{y}_t + \frac{1}{2}y_t^2$$

From this we can derive

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

We are finding $\frac{\partial E_t}{\partial V_t}$

Therefore, because the downward arrow from $\hat{y}_t$ is approaching V, we utilise this.

$$\frac{\partial E_t}{\partial V_t} = \frac{\partial \hat{y}_t}{\partial V_t} \cdot \frac{\partial E_t}{\partial \hat{y}_t}$$

Using basic differentiation,

$$\frac{\partial \hat{y}_t}{\partial V} = h_t V = h_t$$

$$\frac{\partial E_t}{\partial V_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot h_t$$

Remember, we derived $\frac{\partial E_t}{\partial \hat{y}_t}$,

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

$$\frac{\partial E_t}{\partial V_t} = (\hat{y}_t - y_t) \cdot {h_t}^T$$

We are finding $\frac{\partial E_t}{\partial c_t}$

$$\frac{\partial E_t}{\partial c_t} = \frac{\partial \hat{y}_t}{\partial c_t} \cdot \frac{\partial E_t}{\partial \hat{y}_t}$$

Using basic differentiation,

$$\hat{y}_t = h_t \cdot V + c$$

$$\frac{\partial \hat{y}_t}{\partial c} = 1$$

$$\frac{\partial E_t}{\partial V_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot 1$$

Remember, we derived $\frac{\partial E_t}{\partial \hat{y}_t}$,

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

$$\frac{\partial E_t}{\partial c_t} = (\hat{y}_t - y_t)$$

We are finding $\frac{\partial E_t}{\partial h_t}$

Therefore, because the downward arrow from $\hat{y}_t$ is approaching h_t we utilise this.

$$\frac{\partial E_t}{\partial h_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} + \partial h_{next}$$

Where (for simplicity)

$$\partial h_{next} = (\frac{\partial E_t}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t})$$

Or more accurately put (take into account all the h values ahead):

$$\partial h_{next} = \left(\frac{\partial E_t}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t}\right) + \left(\frac{\partial E_t}{\partial h_{t+2}} \cdot \frac{\partial h_{t+2}}{\partial h_t}\right) + \cdots = \sum_{m=1}^{\infty} \left(\frac{\partial E_t}{\partial h_{t+m}} \cdot \frac{\partial h_{t+m}}{\partial h_t}\right)$$

It is important to note that during backpropagation in an RNN, we calculate gradients by propagating backwards through time, starting from the most recent hidden state h_t and moving step-by-step to earlier states (h_{t+1}, h_{t+2}…).

Each hidden state h_t not only contributes to the current output but also influences all future hidden states (h_{t+1}, h_{t+2}…) through the recurrence relationship. As a result, when computing the gradients with respect to h_t, we must account for both:

- The direct contribution to the current time step (e.g., the loss at t).
- The indirect contributions to future time steps (e.g., how h_t affects h_{t+1}, h_{t+2}…).

This recursive dependency means the gradient at h_t accumulates information from all future states, making it critical to consider the recursive nature of the RNN when calculating gradients.

Refer to the diagram for a visual representation of how gradients flow backward through the network.

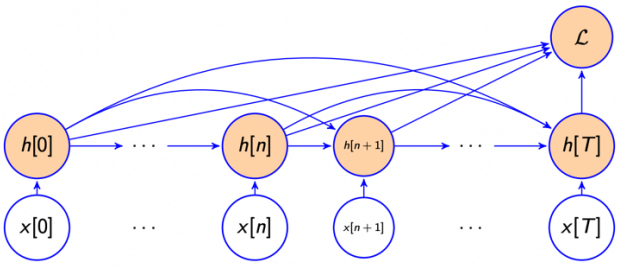


Figure 11: LSTM Back Propagation Through Time Diagram (Johnson, M. H., 2021)

Remember, we derived $\frac{\partial E_t}{\partial \hat{y}_t}$,

$$\frac{\partial E_t}{\partial \hat{y}_t} = \hat{y}_t - y_t$$

Remember,

$$\hat{y}_t = \lambda(V h_t)$$

Important Note: For calculation simplicity, we assume the transformation from h_t to $\hat{y}_t$ performs a linear transformation. But it is important to note that λ could be any suitable function such as softmax for classification tasks or identity for regression, etc.

$$\hat{y}_t = V h_t$$

$$\frac{\partial \hat{y}_t}{\partial h_t} = V$$

Therefore,

$$\frac{\partial E_t}{\partial h_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} + \partial h_{next} = (\hat{y}_t - y_t)V + \partial h_{next}$$

We are finding $\frac{\partial E_t}{\partial C_t}$

$$\frac{\partial E_t}{\partial C_t} = \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial C_t} + \partial C_{next}$$

Where (for simplicity)

$$\partial h_{next} = \left(\frac{\partial E_t}{\partial C_{t+1}} \cdot \frac{\partial C_{t+1}}{\partial h_t}\right)$$

Or more accurately put (take into account all the C values ahead):

$$\partial C_{next} = \left(\frac{\partial E_t}{\partial C_{t+1}} \cdot \frac{\partial C_{t+1}}{\partial h_t}\right) + \left(\frac{\partial E_t}{\partial C_{t+2}} \cdot \frac{\partial C_{t+2}}{\partial h_t}\right) + \cdots = \sum_{m=1}^{\infty} \left(\frac{\partial E_t}{\partial C_{t+m}} \cdot \frac{\partial C_{t+m}}{\partial h_t}\right)$$

Remember we have:

$$h_t = o_t \odot \tanh(C_t)$$

$$\frac{dy}{dx}(\tanh(x)) = 1 - [\tanh(x)]^2$$

$$\frac{\partial h_t}{\partial C_t} = o_t \odot [1 - [\tanh(C_t)]^2]$$

Thus, with this:

$$\frac{\partial E_t}{\partial h_t} = (\hat{y}_t - y_t)V + \partial h_{next}$$

We can derive:

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

We are finding $\frac{\partial E_t}{\partial o_t}$

$$\frac{\partial E_t}{\partial o_t} = \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial o_t}$$

Remember we have:

$$h_t = o_t \odot \tanh(C_t)$$

Thus, we can derive: $\frac{\partial h_t}{\partial o_t}$

$$\frac{\partial h_t}{\partial o_t} = \tanh(C_t)$$

And we have (as derived above):

$$\frac{\partial E_t}{\partial h_t} = \frac{\partial E_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial h_t} + \partial h_{next} = (\hat{y}_t - y_t)V + \partial h_{next}$$

Hence, we have:

$$\frac{\partial E_t}{\partial o_t} = [(\hat{y}_t - y_t)V + \partial h_{next}] \cdot \tanh(C_t)$$

We are finding $\frac{\partial E_t}{\partial b_0}$

To derive the derivative $\frac{\partial E_t}{\partial b_t}$, we must remember the chain rule. The chain rule states that if a variable z depends on y, which in turn depends on x, then the derivative of z with respect to x is:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

So in this context, variable E_t depends on a_0 (which we define as $a_0 = U_o x_t + W_o h_{t-1} + b_o$), which in return depends on b_0.
Thus, we can derive:

$$\frac{\partial E_t}{\partial b_0} = \frac{\partial E_t}{\partial a_0} \cdot \frac{\partial a_0}{\partial b_0}$$

Logically, we can derive $\frac{\partial a_0}{\partial b_0}$:

$$\frac{\partial a_0}{\partial b_0} = 1$$

Hence,

$$\frac{\partial E_t}{\partial b_0} = \frac{\partial E_t}{\partial a_0} \cdot 1$$

$$\frac{\partial E_t}{\partial b_0} = \frac{\partial E_t}{\partial a_0}$$

We are finding $\frac{\partial E_t}{\partial W_0}$

To derive the derivative $\frac{\partial E_t}{\partial W_t}$, we must remember the chain rule. The chain rule states that if a variable z depends on y, which in turn depends on x, then the derivative of z with respect to x is:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

So in this context, variable E_t depends on a_0 (which we define as $a_0 = U_o x_t + W_o h_{t-1} + b_o$), which in return depends on W_0.
Thus, we can derive:

$$\frac{\partial E_t}{\partial W_0} = \frac{\partial E_t}{\partial a_0} \cdot \frac{\partial a_0}{\partial W_0}$$

Logically, we can derive $\frac{\partial a_0}{\partial W_0}$:

$$\frac{\partial a_0}{\partial W_0} = h_{t-1}^{\top}$$

Hence,

$$\frac{\partial E_t}{\partial W_0} = \frac{\partial E_t}{\partial a_0} \cdot h_{t-1}^{\top}$$

We are finding $\frac{\partial E_t}{\partial U_0}$

To derive the derivative $\frac{\partial E_t}{\partial U_t}$, we must remember the chain rule. The chain rule states that if a variable z depends on y, which in turn depends on x, then the derivative of z with respect to x is:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

So in this context, variable E_t depends on a_0 (which we define as $a_0 = U_o x_t + W_o h_{t-1} + b_o$), which in return depends on U_0.
Thus, we can derive:

$$\frac{\partial E_t}{\partial U_0} = \frac{\partial E_t}{\partial a_0} \cdot \frac{\partial a_0}{\partial U_0}$$

Logically, we can derive $\frac{\partial a_0}{\partial U_0}$:

$$\frac{\partial a_0}{\partial U_0} = x_t^{\top}$$

Hence,

$$\frac{\partial E_t}{\partial U_0} = \frac{\partial E_t}{\partial a_0} \cdot x_t^\top$$

We are finding $\frac{\partial E_t}{\partial g_t}$

$$\frac{\partial E_t}{\partial g_t} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial g_t}$$

We know from above:

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

Thus, we derive $\frac{\partial C_t}{\partial g_t}$ from:

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$

Which gets us:

$$\frac{\partial C_t}{\partial g_t} = i_t$$

Thus, we have:

$$\frac{\partial E_t}{\partial g_t} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial g_t} = \frac{\partial E_t}{\partial C_t} \cdot i_t$$

With

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

We are finding $\frac{\partial E_t}{\partial i_t}$

$$\frac{\partial E_t}{\partial i_t} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial i_t}$$

We know from above:

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

Thus, we derive $\frac{\partial C_t}{\partial i_t}$ from:

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$

Which gets us:

$$\frac{\partial C_t}{\partial i_t} = g_t$$

Thus, we have:

$$\frac{\partial E_t}{\partial i_t} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial i_t} = \frac{\partial E_t}{\partial C_t} \cdot g_t$$

With

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

We are finding $\frac{\partial E_t}{\partial W_i}, \frac{\partial E_t}{\partial U_i}, \frac{\partial E_t}{\partial b_i}$ and $\frac{\partial E_t}{\partial W_g}, \frac{\partial E_t}{\partial U_g}, \frac{\partial E_t}{\partial b_g}$

To derive W_i, U_i, b_i and W_g, U_g, b_g, we perform exactly what we did for.
The derivation and calculations are the same.

We reference the equations:

$$g_t = \tanh\left(U_g x_t + W_g h_{t-1} + b_g\right)$$
$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

Remember, we define:

$$a_i = U_i x_t + W_i h_{t-1} + b_i$$

$$a_g = U_g x_t + W_g h_{t-1} + b_g$$

To get:

$$\frac{\partial E_t}{\partial W_i} = \frac{\partial E_t}{\partial a_i} \cdot h_{t-1}^\top$$

$$\frac{\partial E_t}{\partial U_i} = \frac{\partial E_t}{\partial a_i} \cdot x_t^\top$$

$$\frac{\partial E_t}{\partial b_i} = \frac{\partial E_t}{\partial a_i}$$

And

$$\frac{\partial E_t}{\partial W_g} = \frac{\partial E_t}{\partial a_g} \cdot h_{t-1}^\top$$

$$\frac{\partial E_t}{\partial U_g} = \frac{\partial E_t}{\partial a_g} \cdot x_t^\top$$

$$\frac{\partial E_t}{\partial b_g} = \frac{\partial E_t}{\partial a_g}$$

We are finding $\frac{\partial E_t}{\partial f_t}$

$$\frac{\partial E_t}{\partial f_t} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial f_t}$$

We know from above:

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

Thus, we derive $\frac{\partial C_t}{\partial f_t}$ from:

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$

Which gets us:

$$\frac{\partial C_t}{\partial f_t} = C_{t-1}$$

Thus, we have:

$$\frac{\partial E_t}{\partial f_t} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial f_t} = \frac{\partial E_t}{\partial C_t} \cdot C_{t-1}$$

With

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

We are finding $\frac{\partial E_t}{\partial W_f}, \frac{\partial E_t}{\partial U_f}, \frac{\partial E_t}{\partial b_f}$

To derive W_f, U_f, b_f, we perform exactly what we did for.
The derivation and calculations are the same.

We reference the equations:

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

Remember, we define:

$$a_i = U_f x_t + W_f h_{t-1} + b_f$$

To get:

$$\frac{\partial E_t}{\partial W_f} = \frac{\partial E_t}{\partial a_f} \cdot h_{t-1}^{\top}$$

$$\frac{\partial E_t}{\partial U_f} = \frac{\partial E_t}{\partial a_f} \cdot x_t^{\top}$$

$$\frac{\partial E_t}{\partial b_f} = \frac{\partial E_t}{\partial a_f}$$

We are finding $\frac{\partial E_t}{\partial C_{t-1}}$

$$\frac{\partial E_t}{\partial C_{t-1}} = \frac{\partial E_t}{\partial C_t} \cdot \frac{\partial C_t}{\partial C_{t-1}}$$

Remember we have:

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t$$

We can derive $\frac{\partial C_t}{\partial C_{t-1}}$:

$$\frac{\partial C_t}{\partial C_{t-1}} = f_t$$

Thus, with this:

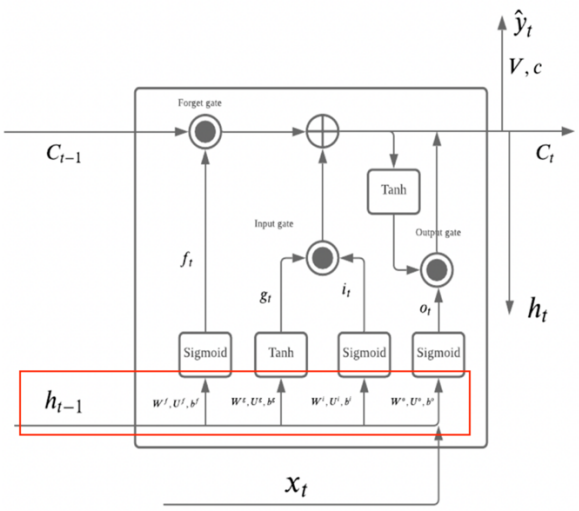
$$\frac{\partial E_t}{\partial C_{t-1}} = \frac{\partial E_t}{\partial C_t} \cdot f_t$$

We can derive:

$$\frac{\partial E_t}{\partial C_t} = [(\hat{y}_t - y_t)V + \partial h_{next}) \cdot o_t \cdot (1 - [\tanh(C_t)]^2)] + \partial C_{next}$$

We are finding $\frac{\partial E_t}{\partial h_{t-1}}$

Because h_{t-1} was used in many different places during the forward pass, we need to collect the gradients. Given an intermediate variable.



We are to now derive the changes from a_i and i_t, g_t, f_t and o_t to determine relationship with E_t. This ensures we are considering the non-linearity in the derivatives.

We will first look at input gate:

$$a_i = U_i x_t + W_i h_{t-1} + b_i$$
$$i_t = \sigma(a_i)$$

$$\frac{\partial E_t}{\partial a_i} = \frac{\partial E_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i}$$

$$\frac{\partial E_t}{\partial i_t} = \frac{\partial E_t}{\partial C_t} \cdot g_t$$

Refer to derivation above to get $\frac{\partial E_t}{\partial i_t}$.

$$\frac{\partial i_t}{\partial a_i} = i_t \odot (1 - i_t)$$

Reference the Appendix 17.2 for proof of derivative of $\frac{\partial i_t}{\partial a_i}$.

$$\frac{\partial E_t}{\partial a_i} = \frac{\partial E_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i} = \left[\frac{\partial E_t}{\partial C_t} \cdot g_t\right] \cdot [i_t \odot (1 - i_t)]$$

Now we look at the candidate value:

$$a_g = U_g x_t + W_g h_{t-1} + b_g$$
$$g_t = \tanh(a_g)$$

$$\frac{\partial E_t}{\partial a_g} = \frac{\partial E_t}{\partial g_t} \cdot \frac{\partial g_t}{\partial a_g}$$

$$\frac{\partial E_t}{\partial g_t} = \frac{\partial E_t}{\partial C_t} \cdot i_t$$

Refer to derivation above to get $\frac{\partial E_t}{\partial g_t}$.

$$\frac{\partial g_t}{\partial a_g} = 1 - \left(\tanh(a_g)\right)^2$$

$$\frac{\partial E_t}{\partial a_g} = \frac{\partial E_t}{\partial g_t} \cdot \frac{\partial g_t}{\partial a_o} = \left[\frac{\partial E_t}{\partial C_t} \cdot i_t\right] \cdot \left[1 - \left(\tanh(a_g)\right)^2\right]$$

Now we look at the forget gate:

$$a_f = U_f x_t + W_f h_{t-1} + b_f$$
$$f_t = \sigma(a_f)$$

$$\frac{\partial E_t}{\partial a_f} = \frac{\partial E_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f}$$

$$\frac{\partial E_t}{\partial f_t} = \frac{\partial E_t}{\partial C_t} \cdot C_{t-1}$$

Refer to derivation above to get $\frac{\partial E_t}{\partial f_t}$.

$$\frac{\partial f_t}{\partial a_f} = f_t \odot (1 - f_t)$$

Reference the Appendix 17.2 for proof of derivative of $\frac{\partial f_t}{\partial a_f}$.

$$\frac{\partial E_t}{\partial a_f} = \frac{\partial E_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f} = \left[\frac{\partial E_t}{\partial C_t} \cdot C_{t-1}\right] \cdot [f_t \odot (1 - f_t)]$$

Now we look at the output gate:

$$a_o = U_o x_t + W_o h_{t-1} + b_o$$
$$o_t = \sigma(a_o)$$

$$\frac{\partial E_t}{\partial a_o} = \frac{\partial E_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial a_o}$$

$$\frac{\partial E_t}{\partial o_t} = \frac{\partial E_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial o_t} = \frac{\partial E_t}{\partial h_t} \cdot \tanh(C_t)$$

$$\frac{\partial h_t}{\partial o_t} = \tanh(C_t)$$

Refer to derivation above to get $\frac{\partial E_t}{\partial o_t}$.

$$\frac{\partial o_t}{\partial a_o} = o_t \odot (1 - o_t)$$

Reference the Appendix 17.2 for proof of derivative of $\frac{\partial o_t}{\partial a_o}$.

$$\frac{\partial E_t}{\partial a_o} = \frac{\partial E_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial a_o} = \left[\frac{\partial E_t}{\partial h_t} \cdot \tanh(C_t)\right] \cdot [o_t \odot (1 - o_t)]$$

Given that we can derive the following:

$$\frac{\partial a_o}{\partial h_{t-1}} = W_o$$

$$\frac{\partial a_f}{\partial h_{t-1}} = W_f$$

$$\frac{\partial a_i}{\partial h_{t-1}} = W_i$$

$$\frac{\partial a_g}{\partial h_{t-1}} = W_g$$

We can write:

$$\frac{\partial E_t}{\partial h_{t-1}} = \frac{\partial E_t}{\partial a_o} \cdot \frac{\partial a_o}{\partial h_{t-1}} + \frac{\partial E_t}{\partial a_f} \cdot \frac{\partial a_f}{\partial h_{t-1}} + \frac{\partial E_t}{\partial a_i} \cdot \frac{\partial a_i}{\partial h_{t-1}} + \frac{\partial E_t}{\partial a_g} \cdot \frac{\partial a_g}{\partial h_{t-1}}$$

$$\frac{\partial E_t}{\partial h_{t-1}} = \sum_{p \in \{o,f,i,g\}} \frac{\partial E_t}{\partial a_p} \cdot \frac{\partial a_p}{\partial h_{t-1}}$$

Or more properly:

$$\frac{\partial E_t}{\partial h_{t-1}} = \sum_{p \in \{o,f,i,g\}} \frac{\partial E_t}{\partial a_p} \cdot W_p$$

$$\frac{\partial E_t}{\partial h_{t-1}} = \frac{\partial E_t}{\partial a_o} \cdot \frac{\partial a_o}{\partial h_{t-1}} + \frac{\partial E_t}{\partial a_f} \cdot \frac{\partial a_f}{\partial h_{t-1}} + \frac{\partial E_t}{\partial a_i} \cdot \frac{\partial a_i}{\partial h_{t-1}} + \frac{\partial E_t}{\partial a_g} \cdot \frac{\partial a_g}{\partial h_{t-1}}$$

Given that $\frac{\partial E_t}{\partial a_p}$ relies on p_t, we can write in more detail (utilising all changes for each of the variables):

$$\sum_{p \in \{o,f,i,g\}} \frac{\partial E_t}{\partial p_t} \cdot \frac{\partial p_t}{\partial a_p} \cdot \frac{\partial a_p}{\partial h_{t-1}}$$

$$\frac{\partial E_t}{\partial h_{t-1}} = \frac{\partial E_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial a_o} \cdot \frac{\partial a_o}{\partial h_{t-1}} + \frac{\partial E_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f} \cdot \frac{\partial a_f}{\partial h_{t-1}} + \frac{\partial E_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i} \cdot \frac{\partial a_i}{\partial h_{t-1}} + \frac{\partial E_t}{\partial g_t} \cdot \frac{\partial g_t}{\partial a_g} \cdot \frac{\partial a_g}{\partial h_{t-1}}$$

Thus, we get:

$$\frac{\partial E_t}{\partial h_{t-1}} = \frac{\partial E_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial a_o} \cdot W_o + \frac{\partial E_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f} \cdot W_f + \frac{\partial E_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i} \cdot W_i + \frac{\partial E_t}{\partial g_t} \cdot \frac{\partial g_t}{\partial a_g} \cdot W_g$$

We can write in this way:

$$\frac{\partial E_t}{\partial h_{t-1}} = \sum_{p \in \{o,f,i,g\}} \frac{\partial E_t}{\partial p_t} \cdot \frac{\partial p_t}{\partial a_p} \cdot W_p$$

We are finding $\frac{\partial E_t}{\partial x_t}$

Very similar to the method above for calculating h_{t-1}.

$$\frac{\partial E_t}{\partial x_t} = \frac{\partial E_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial a_o} \cdot \frac{\partial a_o}{\partial x_t} + \frac{\partial E_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f} \cdot \frac{\partial a_f}{\partial x_t} + \frac{\partial E_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i} \cdot \frac{\partial a_i}{\partial x_t} + \frac{\partial E_t}{\partial g_t} \cdot \frac{\partial g_t}{\partial a_g} \cdot \frac{\partial a_g}{\partial x_t}$$

$$\frac{\partial a_o}{\partial x_t} = U_o$$

$$\frac{\partial a_f}{\partial x_t} = U_f$$

$$\frac{\partial a_i}{\partial x_t} = U_i$$

$$\frac{\partial a_g}{\partial x_t} = U_g$$

Thus, we get:

$$\frac{\partial E_t}{\partial x_t} = \frac{\partial E_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial a_o} \cdot U_o + \frac{\partial E_t}{\partial f_t} \cdot \frac{\partial f_t}{\partial a_f} \cdot U_f + \frac{\partial E_t}{\partial i_t} \cdot \frac{\partial i_t}{\partial a_i} \cdot U_i + \frac{\partial E_t}{\partial g_t} \cdot \frac{\partial g_t}{\partial a_g} \cdot U_g$$

$$\frac{\partial E_t}{\partial x_t} = \sum_{p \in \{o,f,i,g\}} \frac{\partial E_t}{\partial p_t} \cdot \frac{\partial p_t}{\partial a_p} \cdot U_p$$

13. HYBRID MODEL

13.1 Integrating ARMA & LSTM

The code implementation begins by pre-processing stock data, splitting it into training and testing sets, and selecting ARIMA models based on various orders of p, d, and q. It utilises the Minimum Message Length (MML) criterion alongside traditional methods like AIC (Akaike Information Criterion) to select the best ARIMA parameters. This adheres closely to the report's emphasis on MML as a robust model selection tool. The residuals from the ARIMA predictions are then scaled and passed to an LSTM model to capture the nonlinear dependencies not addressed by ARIMA, which aligns with the report's claim that LSTMs can model complexities beyond linear structures. The code mirrors the report's hybrid framework where the final prediction is the sum of the ARIMA predictions and LSTM-modelled residuals.

One key similarity is the use of MML to select ARIMA orders and the training of an LSTM with three layers to predict residuals. Both approaches involve decomposing time series data into linear (ARIMA) and nonlinear (LSTM) components. Additionally, the input structure for the LSTM and the emphasis on residuals as input to capture unexplained dependencies directly follow the report's methodology.

However, the code diverges in several ways:

- Residual Scaling and Slicing: The code uses a MinMaxScaler to normalise residuals before training the LSTM, ensuring all inputs fall within a specific range. This step is not explicitly described in the report.
- Model Selection Implementation: While the report integrates MML for model selection conceptually, the code directly computes the MML using the Fisher Information matrix, log-likelihood, and lattice constant. This shows a more hands-on computational approach compared to the report's theoretical emphasis.
- Data Preparation: The code dynamically adjusts train_len and splits the dataset based on a calculated proportion, while the report focuses more on the theoretical foundation of splitting and not the explicit mechanics of data slicing.
- LSTM Architecture and Training: The code defines the LSTM model with three layers, uses a mean squared error loss function, and trains it for 10 epochs. This implementation detail follows the general structure of Algorithm 1 in the report but uses TensorFlow/Keras for LSTM instead of abstract pseudocode.
- Limited Simulations: The implementation only performs one simulation due to limited computational resources. This differs from the report, which discusses performance results derived from multiple simulations. The lack of repetitions may reduce the robustness and generalisability of the observed outcomes.
- Rolling Forecasting with LSTM: Unlike the report, the code does not perform rolling forecasting with the LSTM. Instead, it uses a static test set, which may miss out on capturing the iterative, time-dependent dynamics of residual forecasting discussed conceptually in the report.
- Fisher Information Matrix Calculation: The Hessian matrix is used to derive the Fisher Information matrix, a critical component of the MML computation. This step introduces potential numerical instability or inaccuracies, especially for large datasets or poorly conditioned models, which the report does not discuss in detail.
- Statsmodels ARIMA Model Limitations: The code uses the Statsmodels ARIMA implementation, which relies on full Maximum Likelihood Estimation (MLE) via the Kalman filter. While robust, this approach may introduce computational inefficiencies and is not explicitly addressed in the report, which assumes ARIMA model outputs are available without discussing underlying computational choices.

These differences may influence the scale and magnitude in numerical results but did not alter the fundamental patterns and relationships identified.

13.2 Algorithmic Implementation

The process begins with loading the dataset (MMM_full.csv) and defining constants such as the test period (T) and lookback window for LSTM. The training length is dynamically calculated to ensure there is sufficient data for training both ARIMA and LSTM models while maintaining the desired test period size. The dataset is then prepared by isolating each stock (ignoring a potential Date column) for individual analysis.

To facilitate the process, helper functions are defined. The split_data function prepares overlapping sequences of time series data for training and testing the LSTM, ensuring that inputs and labels are aligned with the lookback window. Additionally, the calculate_mml function computes the Minimum Message Length (MML), a Bayesian information-theoretic criterion used to select the best ARIMA model by balancing model complexity and fit.

For each stock, the ARIMA model is trained first. A grid search is performed over possible ARIMA orders (p,d,q), with d fixed at 1 (as per the report sheet). For each combination, an ARIMA model is fitted, and various model selection metrics, including MML, are computed. The order with the lowest MML score is selected as the best model. The chosen ARIMA model is then refitted to the training data, and forecasts are generated for the test period. Residuals, which represent the difference between actual values and ARIMA predictions, are calculated and saved for further processing.

The residuals are then scaled using a Min-Max Scaler to normalise them to a range of $[-1,1]$, ensuring that they are suitable for LSTM training. The split_data function is used to prepare input sequences and labels from these scaled residuals. An LSTM model is defined with three layers of 30 units each, where the first two layers pass sequences to the next layer (return_sequences=True), followed by a dense layer that outputs a single prediction. The model is compiled with the Adam optimizer and MSE loss function, and it is trained for 10 epochs with a batch size of 32.

The trained LSTM model is used to predict residuals for the test set, which are then rescaled back to their original range. These residual predictions are added to the ARIMA predictions to form the final hybrid predictions. The performance of this hybrid model is visualised by plotting the actual values, ARIMA predictions, and hybrid predictions. RMSE is calculated for the hybrid predictions and stored for each stock.

This process is repeated for all stocks in the dataset, and the average RMSE across all stocks is computed and printed. The hybrid approach leverages ARIMA for capturing linear dependencies and LSTM for modelling nonlinear residual patterns, combining the two to achieve improved forecasting accuracy. The use of MML for ARIMA model selection is a key differentiator, as it emphasises a balance between model complexity and fit, ensuring robust hybrid model performance.

14. RESULTS

13.1. COMPARISON

Average of RMSE

	ARIMA				ARIMA-LSTM			
	AIC	BIC	HQ	MML87	AIC	BIC	HQ	MML87
T = 3	2.206	**1.984**	2.087	2.309	7.201	**6.863**	6.908	6.940
T = 5	2.256	**2.073**	2.152	2.295	6.167	**6.050**	6.471	6.298
T = 10	2.752	2.759	**2.750**	2.756	5.119	4.992	5.030	**4.904**
T = 30	4.743	4.824	4.761	**4.630**	8.717	8.886	9.074	**8.558**
T = 50	8.737	8.804	8.743	**8.516**	8.789	**8.441**	8.679	8.734
T = 70	12.913	13.101	13.027	**12.563**	11.742	11.644	11.555	**11.500**
T = 100	17.834	18.107	18.026	**17.315**	**18.714**	19.373	19.440	19.770
T = 130	22.875	23.232	23.148	**22.309**	25.830	**25.662**	27.101	27.007
T = 150	25.716	26.152	26.065	**25.073**	27.202	27.223	26.732	**26.657**
T = 200	31.237	31.774	31.684	**30.420**	28.374	29.246	28.849	**28.816**

Figure 12: Derived Results from Integrated ARIMA-LSTM Model

Average of RMSE

	ARIMA				ARIMA-LSTM			
	AIC	BIC	HQ	MML87	AIC	BIC	HQ	MML87
T = 3	**2.987**	3.027	2.914	3.075	4.414	4.302	4.375	**4.289**
T = 5	**4.024**	4.077	4.126	4.163	4.024	3.966	4.081	**3.907**
T = 10	4.748	**4.747**	4.712	4.868	5.359	5.261	5.272	**5.249**
T = 30	5.872	**5.867**	5.910	5.994	5.754	**5.628**	5.726	5.643
T = 50	7.834	7.609	7.726	**6.659**	**7.328**	7.411	7.405	7.384
T = 70	9.991	9.909	10.024	**9.645**	10.393	10.221	10.420	**10.085**

T = 100	14.465	13.991	14.197	**9.866**		9.304	**9.087**	9.235	9.253
T = 130	14.482	14.301	17.672	**13.551**		**13.768**	13.811	13.900	14.581
T = 150	22.985	22.985	23.021	**18.045**		17.778	17.526	17.980	**17.461**
T = 200	31.144	30.502	30.712	**30.286**		26.831	**26.424**	26.662	26.507

Figure 13: ARIMA-LSTM Model Results from Academic Report (Fang et al., 2021)

Similarities

- **General RMSE Trends Across Forecasting Horizons:** Both the report and derived results show an overall increase in RMSE values as the forecasting horizon (T) lengthens. This trend is consistent with expectations, as larger forecasting horizons introduce greater uncertainty, leading to less accurate predictions.
- **MML Superiority in ARIMA Model Selection:** In both cases, MML consistently achieves lower RMSE values for ARIMA models compared to AIC, BIC, and HQ, especially for longer forecasting horizons (e.g., T = 200).
- **ARIMA-LSTM Combined RMSE Trends**: Both results demonstrate that ARIMA-LSTM does not always outperform standalone ARIMA for shorter horizons (T = 3 and T = 5). However, for certain cases (e.g., intermediate horizons like T = 50 and T = 70), ARIMA-LSTM shows competitive or improved performance in both implementations.
- **Performance Stability of MML in ARIMA-LSTM**: For ARIMA-LSTM, MML consistently performs reasonably well compared to the other criteria, particularly for longer horizons. Both sets of results indicate MML as a reliable choice.

Differences

- **Higher ARIMA-LSTM RMSE In Derived Results:** Conversely, the derived ARIMA-LSTM RMSE values are significantly higher for shorter horizons compared to those in the report. For instance, at T = 3, the derived RMSE values for ARIMA-LSTM range from 6.86 to 7.20, whereas the report's values are approximately 4.30 to 4.41. This discrepancy could stem from differences in LSTM implementation, such as architecture adjustments, hyperparameter settings, or residual scaling techniques.

Overall, both sets of results align in highlighting MML as a strong criterion for ARIMA model selection, with consistent trends of increasing RMSE across larger horizons. However, the magnitude of RMSE values differs, with the derived ARIMA implementation achieving better results but ARIMA-LSTM underperforming compared to the report. These differences are likely due to methodological variations, such as scaling, specific architecture implementation, lack of computational power to run simulations, and the absence of rolling forecasting in derived implementation. These findings emphasise the importance between theoretical frameworks and practical implementation choices in determining model performance.

15. CONCLUSION

The report begins with a foundational discussion on non-stationary time series, differentiating key concepts like random walks and the necessity of differencing to achieve stationarity. I explored the mathematical derivations and proofs of differencing, including first, second, and seasonal differencing, to establish the groundwork for ARIMA models. Building upon this, the derivations of AR, MA, and ARMA models were mathematically presented, emphasising their compact forms and integration into seasonal ARIMA (SARIMA) models. These detailed proofs demonstrate a deep understanding of how seasonal and non-seasonal components interplay in time series data.

A significant portion of the report focuses on the theoretical foundations of statistical modelling. This section systematically introduces Maximum Likelihood Estimation (MLE), the score function, Fisher information, and various matrix computations (Hessian and Jacobian), providing the mathematical tools necessary for understanding information criteria such as AIC, BIC, and the unique MML criterion. Through detailed proofs of AIC and BIC, I show their importance in model selection, particularly as benchmarks against MML.

In the MML section, I went beyond a surface-level understanding, dissecting the criterion into its two components: model complexity and goodness of fit. By breaking down the derivation and exploring its Bayesian theories, I highlighted the theoretical strengths of MML compared to AIC and BIC. I also further explored alternative formulations of the MML equation and extended the analysis with additional research into its applications. This focus underscores MML's unique value in selecting ARIMA model parameters and optimising hybrid models.

The deep learning section provided a detailed breakdown of Recurrent Neural Networks (RNNs) and LSTMs, including forward and backward propagation with full mathematical derivations. By thoroughly explaining the architecture and mechanics of LSTMs, including gate functions and gradient computations, I demonstrated the suitability of LSTMs for modelling complex residual patterns in time series data. These derivations serve as a critical foundation for understanding the hybrid ARIMA-LSTM model.

The integration of ARIMA and LSTM was discussed algorithmically, aligning closely with the pseudocode presented in the report. The implementation emphasised residual scaling, training data preparation, and LSTM architecture,

ensuring strict procedure to the report's framework while making necessary adjustments to fit modern libraries like PyTorch and TensorFlow. This implementation allowed for a detailed comparison between the derived results and the report's findings.

The results section compared the RMSE values of ARIMA and ARIMA-LSTM models across various forecasting horizons. While the derived results closely aligned with the report in terms of trends and relationships, minor discrepancies in magnitude highlighted the impact of differences in implementation, such as scaling techniques, LSTM architecture, and computational constraints. Despite these differences, the overarching patterns confirmed the effectiveness of the hybrid model and the reliability of MML as a model selection criterion.

In conclusion, this report demonstrates a detailed approach to understanding, implementing, and evaluating hybrid ARIMA-LSTM models for time series forecasting. By combining mathematical derivations, theoretical insights, and practical experimentation, I have validated the strengths and limitations of the hybrid model. The results show the importance of combining linear and nonlinear models to capture complex time series dependencies and highlight MML's superiority as an information criterion. This work not only replicates but also builds upon the academic report, providing a deeper understanding of hybrid modelling and its potential applications in time series analysis.

16. REFERENCES

Allison, L. (2024). *Fisher Information*. Allison. https://allisons.org/ll/MML/Notes/Fisher/

Arat, M. M. (2019). *Backpropagation Through Time for Recurrent Neural Network*. Mustafa Murat ARAT. https://mmuratarat.github.io/2019-02-07/bptt-of-rnn

Borne, K. (2016). *Unsupervised Anomaly Detection in Sequences Using Long Short Term Memory Recurrent Neural Networks*. ResearchGate. https://www.researchgate.net/publication/326998573_Unsupervised_Anomaly_Detection_in_Sequences_Using_Long_Short_Term_Memory_Recurrent_Neural_Networks

Box, G., Reinsel, G., Ljung, G., & Jenkins, G. (2019). *Time Series Analysis Forecasting and Control*. (5th ed.). Wiley.

Cavanaugh, J., Neath, A. (2018). *The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements*. Wiley. https://iowabiostat.github.io/research-highlights/joe/Cavanaugh_Neath_2019.pdf

Cavanaugh, J. (n.d.). *Unifying the Derivations for the Akaike and Corrected Akaike Information Criteria*. Department of Statistics University of Missouri. https://myweb.uiowa.edu/cavaaugh/doc/pub/aicaicc.pdf

Chen, M. (2022, June). *Vanishing Gradient Problem in training Neural Networks*. Australian National University. https://openresearch-repository.anu.edu.au/server/api/core/bitstreams/ac9d83e7-8ea4-4db4-a6d0-5de511990b0b/content

Chen, Y. (2019). *Lecture 7: Model Selection and Prediction* [Lecture notes]. Site. https://faculty.washington.edu/yenchic/19A_stat535/Lec7_model.pdf

Cincotti, C. (2022). *The Jacobian vs. the Hessian vs. the Gradient*. Carmen's Graphics Blog. https://carmencincotti.com/2022-08-15/the-jacobian-vs-the-hessian-vs-the-gradient/

Do, C. (2008, October 10). *The Multivariate Gaussian Distribution* [Lecture notes]. Site. https://cs229.stanford.edu/section/gaussians.pdf

Fang, Z., Dowe, L. D., Peiris, S. & Rosadi, D. (2021). *Minimum Message Length in Hybrid ARMA and LSTM Model Forecasting*. MDPI. https://www.mdpi.com/1099-4300/23/12/1601

GeeksForGeeks. (2024). *Bayesian Information Criterion (BIC)*. https://www.geeksforgeeks.org/bayesian-information-criterion-bic/

Griffin, M. (2015). *Chapter 3 Selected Basic Concepts in Statistics n Expected Value, Variance, Standard Deviation n Numerical summaries of selected statistics n Sampling*. Slide Player. https://slideplayer.com/slide/4574712/

Johnson, M. H. (2021). *Lecture 23: Recurrent Neural Nets*. The Grainger College of Engineering. https://courses.grainger.illinois.edu/ece417/fa2021/lectures/lec23.pdf

Kovashka, A. (2021). CS 1678: *Intro to Deep Learning Recurrent Neural Networks*. University of Pitssburgh. https://people.cs.pitt.edu/~kovashka/cs1678_sp21/dl_05_rnns.pdf

Krishna, N. (2022). *Backpropagation in RNN Explained*. Toward Data Science. https://towardsdatascience.com/backpropagation-in-rnn-explained-bdf853b4e1c2

Lee, M. J. (n.d.). *Recurrent neural networks and Long-short term memory (LSTM)* [Lecture notes]. Site. https://people.cs.pitt.edu/~jlee/papers/cs3750_rnn_lstm_slides.pdf

Mallya, A. (n.d.). *Backward Pass: Input and Gate Computation - II.* Arun Mallya. https://arunmallya.github.io/writeups/nn/lstm/index.html#/11

Nayak, A. (2020). *Akaike Information Theory.* Toward Data Science. https://towardsdatascience.com/akaike-information-criteria-942d1f554537

Schmidt, D. & Makalic, E. (2008, November 22). *Model Selection Tutorial #1: Akaike's Information Criterion*. Monash University. https://users.monash.edu/~dschmidt/ModelSectionTutorial1_SchmidtMakalic_2008.pdf

Schmidt, D. & Makalic, E. (2022). *Introduction to Minimum Message Length Inference*. Cornell University. https://arxiv.org/abs/2209.14571

Shenoy, K. (2020). *LSTM Back-Propagation — the Math Behind the Scenes. Medium*. https://kartik2112.medium.com/lstm-back-propagation-behind-the-scenes-andrew-ng-style-notations-7207b8606cb2

Stapleton, J. (2012). *Introductory Econometrics Nonstationary Time Series* [Lecture notes]. Site. https://lms.monash.edu/my/

Tae, J. (2020). *Dissecting LSTMs*. Jake Tae. https://jaketae.github.io/study/dissecting-lstm/

Tae, J. (2020). *Fisher Score and Information*. Jake Tae. https://jaketae.github.io/study/fisher/

University of Pennsylvania. (n.d.). *Gaussian Integrals*. https://www.hep.upenn.edu/~johnda/Papers/GausInt.pdf

Wikipedia. (n.d.). *Akaike Information Criterion*. https://en.wikipedia.org/wiki/Akaike_information_criterion

Wikipedia. (n.d.). *Bayesian Information Criterion*. https://en.wikipedia.org/wiki/Bayesian_information_criterion

Wikipedia. (n.d.). *Fisher Information*. https://en.wikipedia.org/wiki/Fisher_information

Wikipedia. (n.d.). *Gaussian Integral*. https://en.wikipedia.org/wiki/Gaussian_integral

Wikipedia. (n.d.). *Generalized Chi Squared Distribution*. https://en.wikipedia.org/wiki/Generalized_chi-squared_distribution

Wikipedia. (n.d.). *The Integral of a Gaussian Function*. https://en.wikipedia.org/wiki/Gaussian_integral#The_integral_of_a_Gaussian_function

Yilidirim, S. (2020). *Expected Value of Random Variables - Explained Simply*. Toward Data Science. https://towardsdatascience.com/expected-value-of-random-variables-explained-simply-a0b02eebd9af

Zhang, M. (2018, October 18). *Time Series: Autoregressive models AR, MA, ARMA, ARIMA* [Lecture notes]. Site. https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class16.pdf

17. APPENDICES

16.1 Taylor Series

What is a Taylor Series?

A Taylor series is a way to represent a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point. This series helps approximate complex functions using polynomials, which are much easier to work with.

Intuition Behind the Taylor Series

The core idea is to express a function f(x) around a point a by considering its value and how it changes near that point. This involves using the function's derivatives at a to build an approximation.

Taylor Series Formula

The Taylor series of a function f(x) around the point x=a is given by:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

Why Use the Taylor Series?

The Taylor series is useful because it allows us to approximate complicated functions using polynomials, which are simpler to analyse and compute.

The approximation becomes more accurate as we include more terms in the series.

16.2 Sigmoid Derivative Proof

We recall that Sigmoid function is:

$$\sigma(x) = \text{sigmoid} = \frac{1}{1+e^{-x}}$$

Hence,

$$\frac{d\sigma}{dx} = \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] = \frac{d}{dx}[(1+e^{-x})^{-1}]$$

We apply the derivative rule (the basic chain rule):

$$\frac{d}{dx}f\big(g(x)\big) = g'(x)\cdot f'(g(x))$$

We let:

$$g(x) = e^{-x} + 1$$

We also remember that:

$$\frac{dy}{dx}\left[e^{f(x)}\right] = f'(x)\cdot e^{f(x)}$$

$$g'^{(x)} = -e^{-x}$$

$$f(x) = (1+e^{-x})^{-1}$$

$$f'^{(x)} = -(1+e^{-x})^{-2}$$

Thus, we get:

$$\frac{d}{dx}f\big(g(x)\big) = -e^{-x}\cdot -(1+e^{-x})^{-2} = e^{-x}\cdot(1+e^{-x})^{-2} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})}\cdot\frac{e^{-x}}{(1+e^{-x})}$$

And remember, we know that $\sigma = \frac{1}{(1+e^{-x})}$, thus we can write:

$$1-\sigma(x) = 1 - \frac{1}{(1+e^{-x})} = \frac{(1+e^{-x})}{(1+e^{-x})} - \frac{1}{(1+e^{-x})} = \frac{e^{-x}}{(1+e^{-x})}$$

$$\frac{d\sigma}{dx} = \sigma(x)\cdot(1-\sigma(x))$$