Technical University of Košice

Faculty of Mining, Ecology, Process Control and Geotechnologies

Advanced prediction models for sales forecasting

Master thesis

2023 **Bc. Aleš Jandera**

Technical University of Košice

Faculty of Mining, Ecology, Process Control and Geotechnologies

Advanced prediction models for sales forecasting

Master thesis

Study Programme:	Process control of raw materials and material extraction and processing
Field of Study:	Cybernetics
Department:	Faculty of Mining, Ecology, Process Control and Geotechnologies (FBERG)
Supervisor:	doc. Ing. Tomáš Škovránek, PhD.

Košice 2023 **Bc. Aleš Jandera**

Abstract in English

Sales forecasting can be divided into two main categories: short-term and long-term forecasting. Short-term forecasting is generally done on a weekly or monthly basis. Long-term forecasting is done on a quarterly or annual basis. There are many different methods that can be used for sales forecasting. The most common method is trend analysis. Trend analysis looks at past sales data to identify patterns and trends that can be used to predict future sales. Other methods include regression analysis and time series analysis. Advance prediction modeling is a type of long-term forecasting. It uses historical data and statistical techniques to predict future sales. Advance prediction modeling is often used by companies to make strategic decisions about inventory, pricing, and marketing.

Keywords in English

Mathematic modeling, forecasting, linear prediction

Abstract in Slovak

Prognózy predaja možno rozdeliť do dvoch hlavných kategórií: krátkodobá a dlhodobá prognóza. Krátkodobé prognózy sa vo všeobecnosti vykonávajú týždenne alebo mesačne. Dlhodobé prognózy sa robia štvrťročne alebo ročne. Existuje mnoho rôznych metód, ktoré možno použiť na predpovedanie predaja. Najbežnejšou metódou je analýza trendov. Analýza trendov sa zameriava na údaje o minulých predajoch, aby identifikovala vzory a trendy, ktoré možno použiť na predpovedanie budúceho predaja. Ďalšie metódy zahŕňajú regresnú analýzu a analýzu časových radov. Predbežné predikčné modelovanie je typ dlhodobého predpovedania. Na predpovedanie budúceho predaja využíva historické údaje a štatistické techniky. Pokročilé predikčné modelovanie často používajú spoločnosti na strategické rozhodnutia o zásobách, cenách a marketingu.

Keywords in Slovak

Matematicke modelovanie, predpoved, linearna predikcia

Bibliographic Citation

JANDERA, Aleš. *Advanced prediction models for sales forecasting*. Košice: Technical University of Košice, Faculty of Mining, Ecology, Process Control and Geotechnologies, 2023. 50s. Supervisor: doc. Ing. Tomáš Škovránek, PhD.

71130

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA BANÍCTVA, EKOLÓGIE, RIADENIA A GEOTECHNOLÓGIÍ

Ústav riadenia a informatizácie výrobných procesov

Z A D A N I E
D I P L O M O V E J P R Á C E

Študijný odbor: **Kybernetika**

Študijný program: **Riadenie procesov získavania a spracovania surovín**

Názov práce:

Pokročilé predikčné modely pre prognózu predaja

Advanced prediction models for sales forecasting

Študent: **Bc. Aleš Jandera**

Školiteľ: **doc. Ing. Tomáš Škovránek, PhD.**

Školiace pracovisko: **Ústav riadenia a informatizácie výrobných procesov**

Konzultant práce:

Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Úvod
2. Analýza súčasného stavu
3. Návrhová časť
4. Zhodnotenie prínosu práce a návrh odporúčaní pre ďalší postup
5. Záver

Odporúčaný rozsah práce: 50 - 70 strán

Jazyk, v ktorom sa práca vypracuje: anglický

Termín pre odovzdanie práce: 21.04.2023

Dátum zadania diplomovej práce: 31.10.2022

.....

Dr. h. c. prof. Ing. Michal Cehlár, PhD.

dekan fakulty

Declaration

I hereby declare that this thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Košice, 21.4.2023 . . . . . .

Signature

Acknowledgement

Very strong thanks to whole teaching staff at the Institute of Control and Informatization of Production Processes for their patience, leadership and knowledge which helped me to write this thesis. I would like to express my deepest appreciation to my supervisor Tomáš Škovránek for his time and advices.

Contents

List of Figures

Introduction

Linear prediction [1] is a method used in signal processing to predict future values of a time series based on past observations. The technique is based on the assumption that the signal can be modeled as a linear combination of past values and a noise term. Long-term linear prediction refers to the application of this method to predict values over a longer period of time, such as months or years. It requires a greater amount of data and is more complex than short-term prediction, but can be useful in areas such as stock market forecasting and weather prediction. The goal of this master thesis is to developing new algorithms and mathematical models to improve the accuracy of long-term predictions in sales forecasting. Matlab [1] livescript [2] will be used as development enviroment.

Task formulation

Proposed a mathematical model and alghoritms for sales forcasting based on long-term prediction with improved Levinson - Durbin scheme which should have better performance and acuraccy than known linear prediction mechanism.

[1] MATLAB is a fourth-generation programminglanguage and numerical analysis environment. Uses for MATLAB include matrix calculations, developing and running algorithms, creating user interfaces (UI) and data visualization.

1 Analytical part

1.1 Introduction

Linear prediction is a statistical method used to predict future values based on historical data. The Durbin-Levinson algorithm is a method for solving the linear prediction problem for autoregressive (AR) models[1], which are models where the current output depends on previous outputs. The algorithm solves the linear prediction problem by finding the coefficients of the AR model that minimize the prediction error. The resulting AR coefficients can be used to make predictions about future values based on past observations. This method should be used with using the pattern of linear relationship between the independent and dependent variables. Here's a basic outline of the steps involved in using linear prediction to forecast sales data:

1. Collect sales data: Gather the historical sales data for the product or service that you want to forecast.
2. Plot the data: Plot the sales data over time to visually inspect the trend and identify any patterns.
3. Choose a model: Select an appropriate linear model to represent the relationship between the independent and dependent variables in the data. For example, you might choose a simple linear regression model.
4. Train the model: Train the selected model on the historical sales data using a method such as least squares regression.

[1] Autoregressive (AR) models are time series models that describe the relationship between the current value of a variable and its past values. In an autoregressive model, each observation is modeled as a linear combination of past observations, with weights called AR coefficients. AR models are widely used in various fields such as economics, engineering, and finance for modeling and forecasting time series data. The order of the AR model, denoted as "p", refers to the number of past values used to predict the current value. For example, an AR(1) model uses only the previous observation to predict the current value, while an AR(2) model uses the previous two observations.

5. Make predictions: Use the trained model to make predictions on future sales data. You may want to generate predictions for several months or years in advance.

6. Evaluate the model: Assess the accuracy of the predictions by comparing them to the actual sales data. Use metrics such as mean absolute error or root mean squared error to quantify the model's performance.

7. Refine the model: If necessary, refine the model by adding additional independent variables or transforming the existing variables. Repeat the training and evaluation steps until you have a model that provides accurate forecasts.

For shift calculation in longterm prediction can be used autocorrelation method, in this thesis neural site for identification shift will be developed and similar mechanism for optimal order detection.

1.2 Models used for sales data

There are several mathematical models used for sales prediction, including:

1. Time series models: These models are used to analyze and forecast sales data over time, such as seasonal patterns, trends, and fluctuations. Examples include ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal ARIMA), and exponential smoothing.

2. Regression models: These models use historical data to determine the relationship between sales and one or more independent variables, such as price, promotion, and advertising. Examples include linear regression, logistic regression, and multiple regression.

3. Decision tree models: These models use a tree-like structure to make decisions based on the relationship between sales and multiple independent variables. Examples include CART (Classification and Regression Tree) and Random Forest.

4. Machine learning models: These models use algorithms such as neural networks and support vector machines to make predictions based on patterns in the data.

The choice of mathematical model depends on the characteristics of the data, the desired level of accuracy, and the computational resources available.

1.2.1 Time-series models

Time-series models are mathematical models used to analyze and forecast data that are collected over time [3]. These models are used to study and make predictions about the trends, patterns, and behavior of the data over time, taking into account historical values and their relationship with the present. Time-series models are widely used in areas such as economics, finance, and weather forecasting, among others. The models are based on various statistical techniques, including ARIMA (AutoRegressive Integrated Moving Average), SARIMA (Seasonal ARIMA), and exponential smoothing, among others. The goal of time-series modeling is to build a mathematical representation of the underlying process that generates the time-series data, allowing for accurate prediction of future values. Time-series models are statistical models used to analyze and make predictions about time-dependent data. They are widely used in various fields, including finance, economics, engineering, and social sciences.
Time-series models make use of past values of a variable to predict future values. They assume that there is a pattern or trend in the data that can be used to forecast future behavior. Some commonly used time-series models include:

1. Autoregressive Integrated Moving Average (ARIMA): This model is used to analyze and forecast stationary time-series data. It consists of three components: autoregression, differencing, and moving average.

2. Seasonal Autoregressive Integrated Moving Average (SARIMA): This model is an extension of ARIMA that takes into account seasonal patterns in the data.

3. Exponential Smoothing (ETS): This model is used to forecast time-series data that has a trend and/or seasonality. It uses a smoothing parameter to assign more or less weight to past observations based on their recency.

4. Vector Autoregression (VAR): This model is used when there are multiple time-series variables that influence each other. It can be used to analyze the relationships between these variables and to make predictions about their future behavior.

5. These models are valuable tools for analyzing and predicting time-series data, but they require careful consideration of the specific characteristics of the data being analyzed and the appropriate model to use.

1.2.2 Regression models

Regression models are a type of statistical models used to examine the relationship between a dependent variable and one or more independent variables [4]. The goal of regression analysis is to model the relationship between these variables and make predictions about the dependent variable based on the values of the independent variables. Regression models are widely used in many fields, including economics, finance, marketing, and social sciences, to make predictions and understand the relationship between variables. There are several types of regression models, including:

1. Linear regression: a simple regression model where the relationship between the dependent and independent variables is modeled using a linear equation.

2. Logistic regression: used for binary classification problems where the dependent variable is binary and the goal is to model the relationship between the independent variables and the probability of the dependent variable being either 0 or 1.

3. Multiple regression: used when there are multiple independent variables and the goal is to model the relationship between all of these variables and the dependent variable.

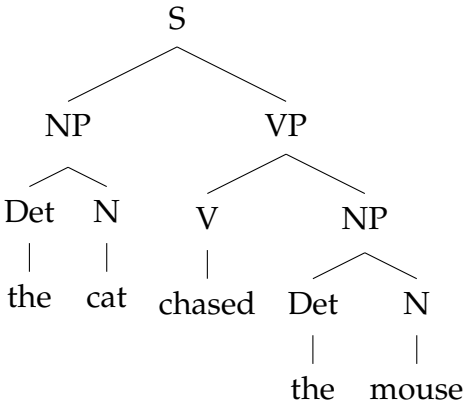
4. Polynomial regression: used when the relationship between the dependent and independent variables is non-linear and can be modeled using a polynomial equation.

The choice of regression model depends on the nature of the data and the research question being asked.

1.2.3 Decision tree models

Decision tree models are a type of machine learning models used for both regression and classification tasks [5]. They are tree-like structures that make predictions by breaking down a dataset into smaller and smaller subsets, based on the values of the input variables. At each internal node of the tree, a decision rule

is used to split the data based on the value of a feature, and the process continues until the data are separated into homogeneous groups, or leaves. The predictions are then made based on the average or majority class in each leaf node.



Decision trees have several advantages, including ease of interpretability, handling of non-linear relationships, and ability to handle both categorical and numerical data. Some examples of decision tree algorithms are CART (Classification and Regression Tree) and Random Forest.
The decision tree model is trained using a dataset, and the tree structure is built using a greedy algorithm that seeks to maximize the reduction in impurity of the target variable at each split. The model can then be used to make predictions on new data by following the decision rules in the tree.

1.2.4 Machine learning models

Machine learning models are a subset of artificial intelligence that allows computers to learn and make predictions or decisions without being explicitly programmed. Machine learning models are based on algorithms that use statistical methods to find patterns in data and make predictions about new, unseen data.
There are several types of machine learning models, including:

1. Supervised learning: where the model is trained on labeled data, with the goal of learning the relationship between the input features and the target variable, and making predictions about the target variable for new, unseen data.

2. Unsupervised learning: where the model is trained on unlabeled data, with the goal of finding patterns or structure in the data, such as clustering or dimensionality reduction.

3. Reinforcement learning: where the model learns by receiving rewards or penalties for its actions in an environment, with the goal of maximizing the

reward over time.

4. Deep learning: a subset of machine learning that uses artificial neural networks with multiple hidden layers to model complex relationships in the data.

The choice of machine learning model depends on the problem being solved and the type of data being used. Machine learning models have been applied to a wide range of tasks, including image and speech recognition, natural language processing, and predictive modeling. Advances in machine learning (ML), faster processors and the availability of digitized healthcare data have contributed to a growing number of papers describing ML applications in healthcare. [6]

1.3 Neural networks

A neural network is a type of machine learning algorithm inspired by the structure and function of biological neurons in the human brain. It is composed of interconnected nodes, called neurons, that are organized into layers. The input layer receives raw data, such as images or text, and passes it on to the hidden layers, which perform calculations and apply weights to the input data to create a prediction. Finally, the output layer produces the final prediction or classification.

As you can see on image 1.1 each input Xn should be properly weighted by a certain weight Wn before all the signals enter the summation stage. Afterwards, the weighted summation is forwarded into the activation unit producing the neuron's output signal.

Neural networks are trained on large datasets using a process called backpropagation, which adjusts the weights and biases of the neurons to minimize the error between the predicted output and the actual output. Once a neural network has been trained, it can be used to make predictions on new data.

A neuron is a basic building block of a neural network, also known as an artificial neuron or a perceptron. It is modeled after the biological neuron in the human brain, which receives input signals from other neurons, processes them, and sends output signals to other neurons.

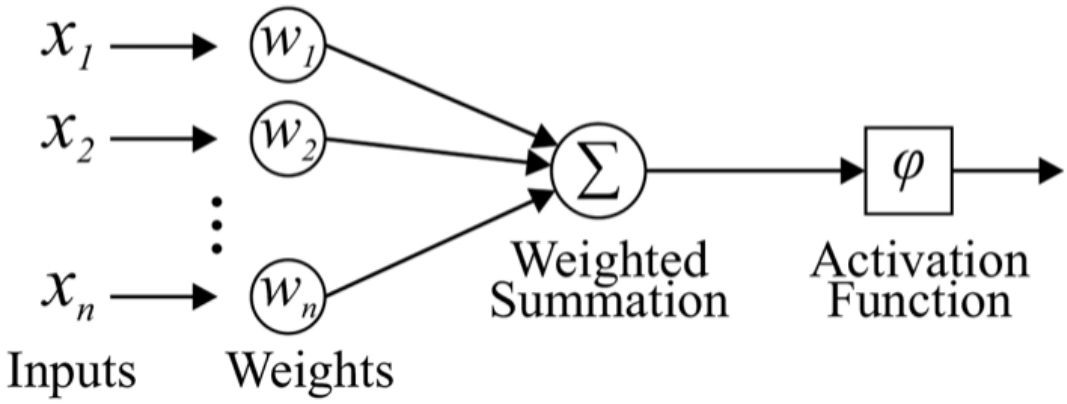


Figure 1.1: Perceptron preview [7]

In a neural network, a neuron receives input from other neurons or directly from the input data, applies a mathematical function to the input, and produces an output that is sent to other neurons in the network. The input to a neuron is usually a vector of numbers, and each input is multiplied by a corresponding weight. The neuron then sums up the weighted inputs, adds a bias term, and applies an activation function to the result.

The purpose of the activation function is to introduce nonlinearity into the neuron, which allows the neural network to learn complex patterns and relationships in the data. There are several different types of activation functions that can be used, such as the sigmoid function, ReLU (Rectified Linear Unit) function, and tanh (hyperbolic tangent) function.

The output of a neuron is typically fed into other neurons in the next layer of the neural network. The weights and biases of the neurons are adjusted during the training process using a technique called backpropagation, which involves computing the gradient of the error with respect to the weights and updating them using an optimization algorithm such as stochastic gradient descent.
Overall, the neurons in a neural network work together to learn patterns and relationships in the input data and produce output that can be used for a variety of tasks, such as classification, regression, and prediction.
Neural networks have been successfully applied in a wide range of fields, including image and speech recognition, natural language processing, and autonomous vehicles, among others.

Neural networks can be broadly classified into the following types:

1. Feedforward Neural Networks: These are the most basic type of neural networks, where the information flows only in one direction, from input to output. These networks can have one or more hidden layers and are often used for classification or regression tasks.

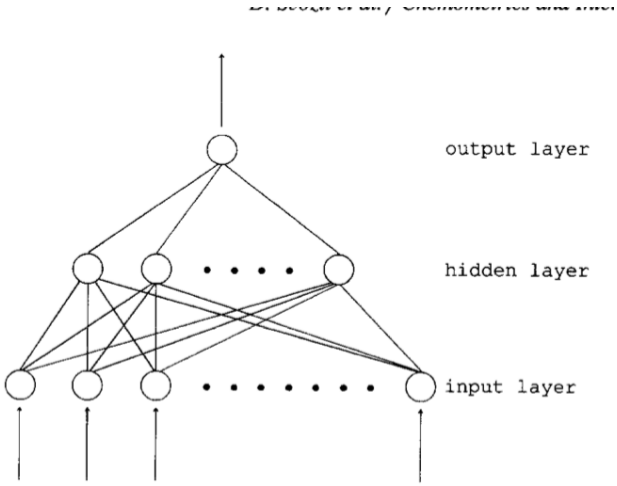


Figure 1.2: Typical feed-forward neural network composed of three layers. [**SVOZIL199743**]

2. Convolutional Neural Networks (CNNs): These networks are specialized for processing images and are commonly used in computer vision tasks. They use convolutional layers to extract features from images and can learn to recognize patterns and objects in images.

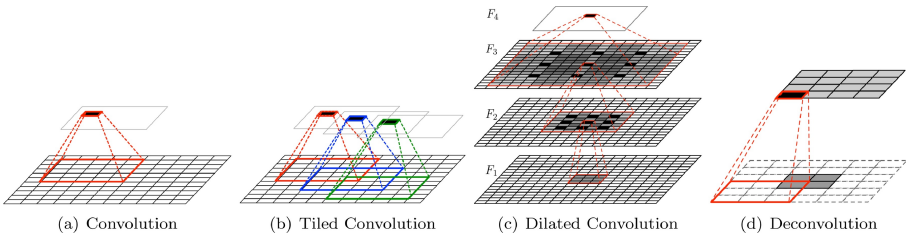


Figure 1.3: Illustration of (a) Convolution, (b) Tiled Convolution, (c) Dilated Convolution, and (d) Deconvolution. [8]

3. Recurrent Neural Networks (RNNs): These networks are designed to work with sequential data, such as time-series or natural language data. They

have loops that allow information to be passed from one time-step to the next, enabling them to capture temporal dependencies in the data.

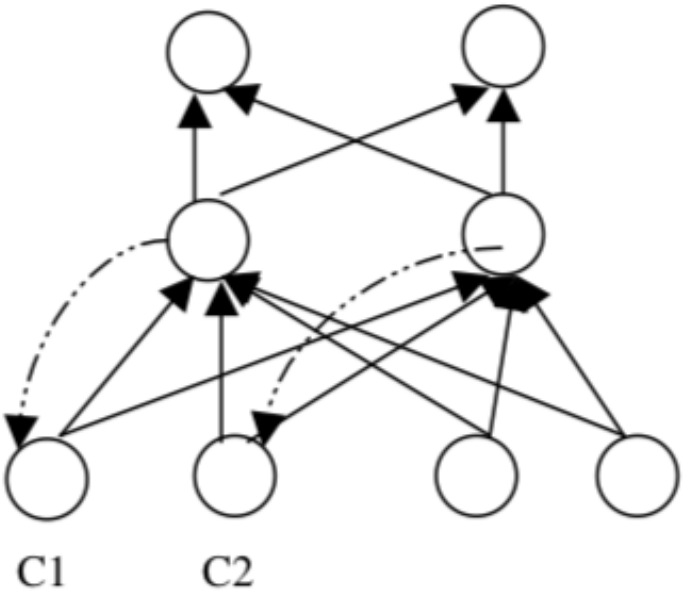


Figure 1.4: Typical recurrent network. [9]

4. Long Short-Term Memory Networks (LSTMs): These are a type of RNN that are designed to address the problem of vanishing gradients in traditional RNNs. They use memory cells and gates to selectively retain or forget information over time, making them well-suited for learning from long sequences.

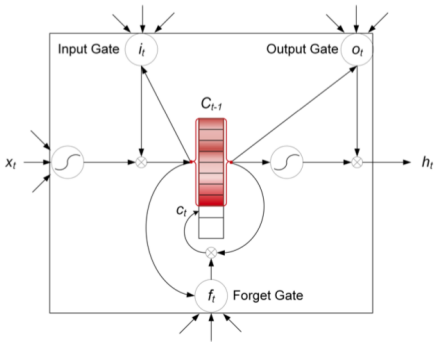


Figure 1.5: Long short-term memory network. Color indicates degree of memory activation. [10]

5. Autoencoder Neural Networks: These networks are used for unsupervised learning and are designed to learn a compressed representation of the input data. They consist of an encoder that maps the input data to a compressed representation, and a decoder that maps the compressed representation back to the original data.

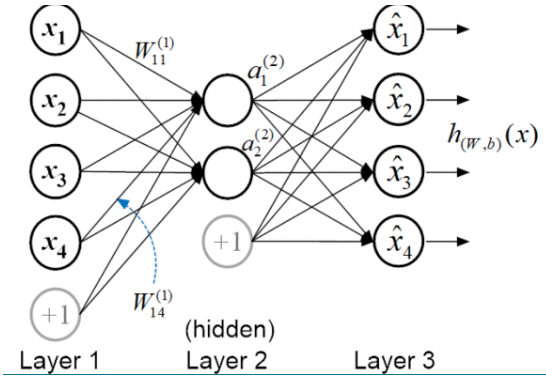


Figure 1.6: An autoencoder neural network. [11]

6. Generative Adversarial Networks (GANs): These networks consist of two networks, a generator and a discriminator, that are trained together in a game-theoretic framework. The generator is trained to generate realistic data samples, while the discriminator is trained to distinguish between real and generated data samples.

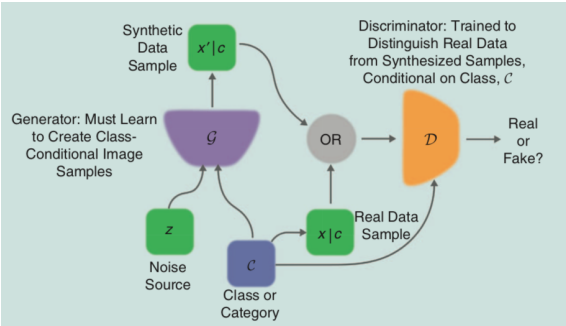


Figure 1.7: the conditional GAN example. [12]

These are some of the most common types of neural networks, but there are many other specialized types of neural networks that have been developed for specific tasks, such as object detection, speech recognition, and natural language processing.

1.3.1 Classification

Neural network data classification is a technique for categorizing data into different classes or categories based on patterns and features present in the data. A neural network is a type of machine learning algorithm that is modeled after the structure and function of the human brain. It is composed of interconnected nodes or neurons that are organized into layers.
In a classification task, the neural network is trained on a dataset that is labeled with the correct class for each example. During training, the network learns to recognize patterns and features in the input data that are associated with each class. The process of training involves adjusting the weights and biases of the neurons in the network to minimize the error between the predicted class and the actual class of each example in the training set.
Once the neural network is trained, it can be used to classify new, unseen examples by inputting the data into the network and obtaining a prediction of the most likely class. The output of the neural network is a probability distribution over the different classes, with the highest probability indicating the predicted class.
Neural network data classification has been successfully applied to a wide range of tasks, including image classification, speech recognition, natural language processing, and fraud detection, among others.

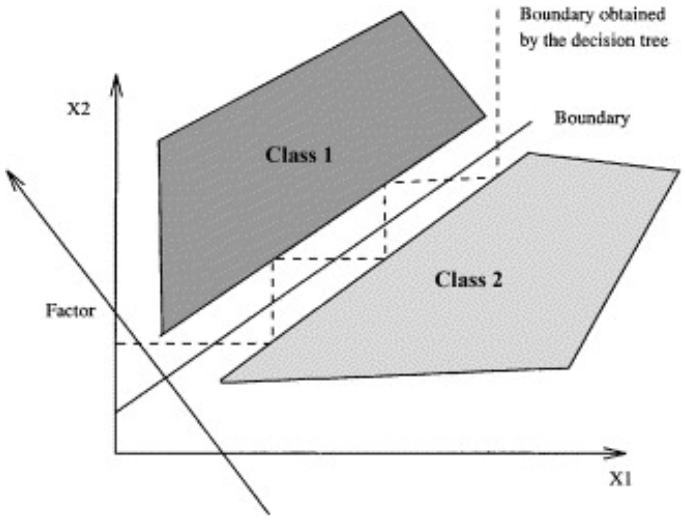


Figure 1.8: The staircase effect: on this toy problem, the true variable is in a one-dimensional factor space obtained by a linear combination of the inputs. The splits produced by the tree can obtain accurate performance in classification, but cannot be used to interpret the classification.[13]

1.3.2 Activation functions

There are several types of activation functions used in neural networks, including:

1. Sigmoid Function: The sigmoid function is a commonly used activation function that maps any input value to a value between 0 and 1. It is typically used in binary classification problems and in the output layer of neural networks that produce probability estimates.

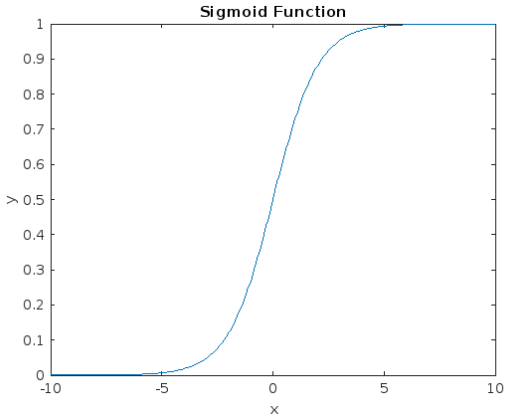


Figure 1.9: Exmaple of Sigmoid function

2. ReLU (Rectified Linear Unit): The ReLU function is another popular activation function that maps any input value less than 0 to 0, and any input value greater than or equal to 0 to the input value itself. It is computationally efficient and has been shown to work well in deep neural networks.

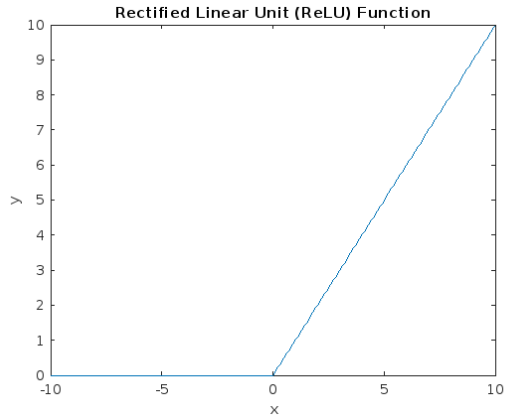


Figure 1.10: Exmaple of ReLU function

3. Tanh Function: The tanh (hyperbolic tangent) function is similar to the sigmoid function, but it maps input values to a range between -1 and 1. It is commonly used in the hidden layers of neural networks.

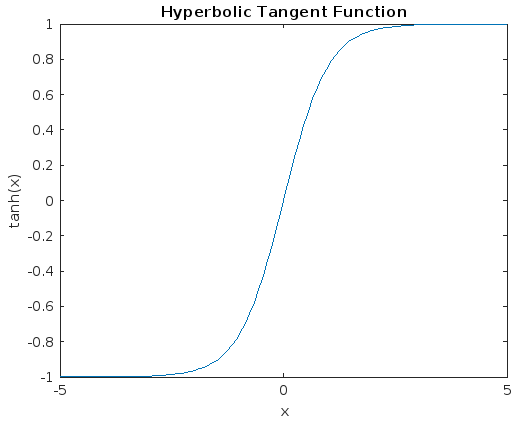


Figure 1.11: Exmaple of Tanh function

4. Softmax Function: The softmax function is often used in the output layer of neural networks that produce multi-class classification predictions. It maps the outputs to a probability distribution over the possible classes.

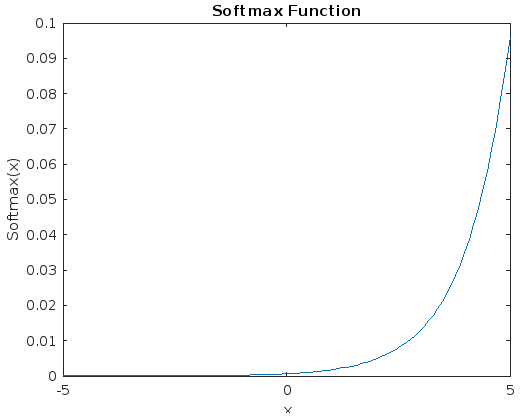


Figure 1.12: Exmaple of Softmax function

5. Leaky ReLU: The Leaky ReLU function is similar to the ReLU function, but it allows a small, non-zero gradient when the input value is negative. This can help to prevent the "dying ReLU" problem, where some ReLU units become inactive and stop contributing to the network's output.

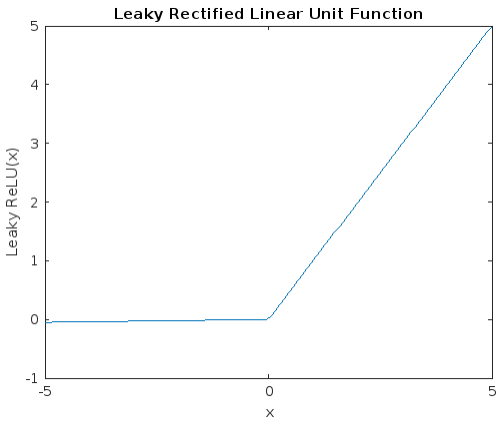


Figure 1.13: Exmaple of Leaky ReLU function

6. ELU (Exponential Linear Unit): The ELU function is similar to the ReLU function, but it allows negative values to have non-zero outputs. This can help to prevent the "dying ReLU" problem and can improve the performance of deep neural networks.

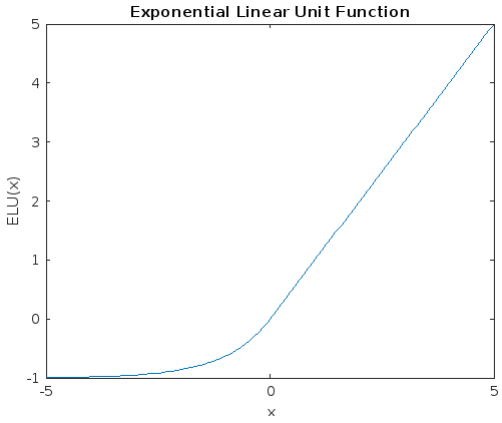


Figure 1.14: Exmaple of ELU function

These are some of the most commonly used activation functions in neural networks, but there are many other types of activation functions that have been developed for specific tasks or to address certain problems.

1.3.3 Network prediction

Neural networks can be used for long-term linear prediction in time-series data, including speech, health, and stock data. Neural networks can be trained to model and forecast the patterns in the data, including shifts in the long-term linear prediction. Neural networks have been shown to be effective in capturing the complex relationships between variables in time-series data, and can learn to identify subtle patterns and trends that may be difficult to detect using traditional statistical methods. In the context of long-term linear prediction, neural networks can be trained on historical data to identify patterns and trends in the data and make predictions for future values. They can also be used to detect shifts in the long-term linear prediction, which can be useful for identifying changes in the underlying processes that generate the data. However, it's important to note that neural networks can be computationally expensive and require large amounts of data to train effectively. They also require careful tuning of hyperparameters and selection of appropriate architecture to achieve good performance. In addition, the interpretation of the results of a neural network can be more challenging than with traditional statistical methods. Overall, neural networks can be a powerful tool for long-term linear prediction in time-series data, including shifts in the data, but their use should be carefully considered based on the specific application and available data.

1.4 Linear prediction

Linear prediction is a statistical technique used to forecast future values based on past observations. It is a method for modeling the relationship between a dependent variable and one or more independent variables in a linear form. The goal of linear prediction is to find the best linear approximation of the relationship between the variables, which can then be used to make predictions about future values of the dependent variable.

Linear prediction can be performed using simple linear regression or multiple linear regression, depending on the number of independent variables involved [14]. In simple linear regression, a single independent variable is used to predict the value of the dependent variable, while in multiple linear regression, multiple independent variables are used to make the prediction.

Linear prediction models are commonly used in finance, economics, and engineering, among other fields, to forecast future values of time series data, such as

stock prices, sales, or demand. The accuracy of linear prediction models depends on several factors, including the quality of the data, the choice of independent variables, and the degree of linearity in the relationship between the variables.

1.4.1 Levinson-Durbin scheme

The Levinson-Durbin algorithm [15] is an iterative numerical method used to solve the autoregressive (AR) model of a time series.

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_t \tag{1.1}$$

where Y_t is the dependent variable at time t, β_0 is the intercept, $\beta_1, \beta_2, \ldots, \beta_p$ are the regression coefficients, $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ are the lagged values of the dependent variable up to order p, and ϵ_t is the error term at time t.

AR models are used to model and forecast time series data [16], such as sales data, by assuming that each future value of the series depends on a linear combination of previous values.

The Levinson-Durbin algorithm solves the AR model by iteratively updating the coefficients of the model to minimize the prediction error between the model and the actual data. The algorithm is fast and efficient, and it is widely used in digital signal processing, speech processing, and control systems, among other applications. The Levinson-Durbin recursion schema is a method for solving the autocorrelation equations of a linear prediction problem. It can be expressed in the form of a triangular system of equations:

$$\begin{aligned}
\alpha_0 &= r_0 \\
\alpha_k &= \frac{1}{k} \sum_{i=1}^{k} a_{k-i} r_i \quad (1 \leq k < p) \\
\alpha_p &= \frac{1}{p} \sum_{i=1}^{p} a_{p-i} r_i,
\end{aligned}$$

where r_i is the ith autocorrelation coefficient, α_k is the kth reflection coefficient, and a_k is the kth prediction coefficient.

The recursion begins with $\alpha_0 = r_0$, and computes each subsequent reflection coefficient α_k in terms of the previous coefficients $\alpha_0, \alpha_1, \ldots, \alpha_{k-1}$ and the autocorrelation coefficients $r_1, r_2, \ldots, r_k$.

The Levinson-Durbin algorithm is often used as an alternative to the Yule-Walker equations, which are another commonly used method for solving AR models. Unlike the Yule-Walker equations, the Levinson-Durbin algorithm can be easily

modified to handle non-stationary time series data, and it is also more robust to numerical issues such as rounding errors.

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} -r_1 \\ -r_2 \\ \vdots \\ -r_p \end{bmatrix},$$

where $r_0, r_1, \ldots, r_{p-1}$ are the autocorrelation coefficients and $a_1, a_2, \ldots, a_p$ are the prediction coefficients.

The algorithm proceeds as follows:

1. Set $\alpha_0 = r_0$ and $k = 1$.

2. Compute the reflection coefficient α_k using the formula

$$\alpha_k = -\frac{1}{\alpha_{k-1}} \sum_{i=1}^{k-1} \alpha_{k-i} r_i.$$

3. Compute the new prediction coefficients $a_1, a_2, \ldots, a_k$ using the formulas

$$\begin{aligned} a_k &= -\alpha_k \\ a_i &= a_{i-1} - \alpha_k a_{k-i} \quad (1 \leq i < k). \end{aligned}$$

4. If $k = p$, terminate. Otherwise, increment k and repeat from step 2.

1.4.2 Short-term linear prediction

Short-term linear prediction [17] refers to the use of linear prediction techniques to make predictions about the near-term future values of a time series. It is used to forecast future values of a dependent variable based on its past values and any relevant independent variables.

In short-term linear prediction, the focus is on accurately predicting the next few values of the dependent variable, typically in the range of several weeks to a few months. The linear prediction models used for short-term forecasting are typically simple and straightforward, often using a small number of independent variables. The goal is to provide a quick and easily interpretable forecast that can

be used to make operational decisions in the short-term.

$$\begin{aligned}
\hat{a}_0 &= 1 \\
\hat{a}_m &= \frac{1}{m}\sum_{j=1}^{m} r_j \hat{a}_{m-j} - \frac{\sum_{j=1}^{m-1} \hat{a}_j r_{m-j+1}}{m} \\
\gamma_m &= (1 - \sum_{j=1}^{m} \hat{a}_j^2) r_0 \\
\hat{E}_m &= \gamma_m - \sum_{j=1}^{m} \hat{a}_j r_j
\end{aligned} \tag{1.2}$$

- $\hat{a}_m$ is the m-th order forward prediction coefficient
- r_j is the j-th autocorrelation coefficient
- γ_m is the m-th reflection coefficient
- $\hat{E}_m$ is the estimated prediction error at order m
- m is the current order of the predictor (typically starts at $m = 0$)

Common techniques for short-term linear prediction include moving average models, exponential smoothing, and autoregressive models. These methods use the historical data of the time series to model the relationship between the dependent and independent variables, and to make predictions about future values. The accuracy of short-term linear predictions can be evaluated using metrics such as mean absolute error, mean squared error, or the correlation coefficient between the actual and predicted values.

1.4.3 Long-term linear prediction

Long-term linear prediction is a technique commonly used in the analysis of time-series data, including health data. In the context of health data, the shift of long-term linear prediction refers to the way in which the patterns in the data change over time, reflecting changes in the underlying health status of the individual or population being studied.

For example, in the case of a patient with a chronic disease, the long-term linear prediction of their health data may show a gradual decline over time as the disease progresses. Alternatively, the data may show periodic shifts corresponding to changes in medication or other interventions. In population health studies, long-term linear prediction can be used to identify trends and patterns in health outcomes over time. For example, a long-term linear prediction model might be

used to track changes in the prevalence of a particular disease or condition over a period of several years, taking into account factors such as demographic changes and changes in healthcare policy. Overall, the shift of long-term linear prediction for health data reflects the dynamic nature of health status and healthcare interventions, and can be a powerful tool for understanding trends and patterns in health outcomes over time.

Long-term linear prediction refers to the use of linear prediction techniques to make predictions about the future values of a time series over an extended period of time, typically several months to several years. Unlike short-term linear prediction, which focuses on forecasting the near-term future, long-term linear prediction aims to provide a more comprehensive and accurate forecast of future values.

$$\hat{y}(t) = \sum_{i=1}^{p} \beta_i y(t-i) \tag{1.3}$$

- $\hat{y}(t)$ is the predicted value of the time series at time t
- $y(t-i)$ is the value of the time series at time t-i (i.e., p lags back)
- $\beta_1, \beta_2, \ldots, \beta_p$ are the coefficients or weights assigned to the past values of the time series

In long-term linear prediction, more sophisticated models [18] are typically used, such as multiple linear regression or time series models, and a larger number of independent variables may be considered. The models are also trained on a larger historical dataset to ensure that they capture any long-term trends or patterns in the data.

Long-term linear prediction is commonly used in fields such as finance, economics, and marketing, to make long-term projections about variables such as sales, demand, or stock prices. The goal is to provide a comprehensive and accurate forecast that can be used to make strategic decisions in the long-term. The accuracy of long-term linear predictions can be evaluated using the same metrics as for short-term linear predictions [19], as well as additional metrics such as mean absolute percentage error or mean absolute scaled error.

In some long-term prediction use cases it needs to solve the suppression of Late Reverberation Effect[2] this can be done by with minimal performance degradation

[2]Late Reverberation Effect refers to the decay of sound in an environment after the initial sound source has stopped. This effect results in the persistence of sound in a space for a short period of time and helps create the characteristic ambiance of a room or space. It is an important aspect of room acoustics and is used in sound design and music production to enhance the perceived sound quality and spatial experience of audio.

by framework developed by Keisuke Kinoshita [20] for both single-channel and multichannel scenarios.

1.4.4 Order of linear prediction

The optimal order of linear prediction refers to the number of past observations that should be used to predict the next observation in a time series. There are several methods for finding the optimal order of linear prediction, including the Akaike Information Criterion (AIC), the Bayesian Information Criterion (BIC), and cross-validation. The Akaike Information Criterion (AIC) is a measure of the relative quality of a statistical model for a given set of data. The AIC score is based on the goodness of fit of the model and the complexity of the model. In the context of linear prediction, the AIC can be used to select the optimal order of the autoregressive (AR) model. The AR model uses past observations to predict the next observation in a time series. The optimal order of the AR model is the order that minimizes the AIC score. The Bayesian Information Criterion (BIC) is similar to the AIC but places a greater penalty on model complexity. The BIC can also be used to select the optimal order of the AR model. Cross-validation is a technique that involves splitting the data into training and testing sets. The model is trained on the training set, and the performance of the model is evaluated on the testing set. The optimal order of the AR model is the order that produces the best performance on the testing set. Overall, the optimal order of linear prediction can be found using a combination of these methods, taking into account the complexity of the model, the goodness of fit, and the performance on a testing set.

Autocorrelation can be a useful tool for detecting the optimal shift in long-term linear prediction for time-series data. Autocorrelation measures the linear relationship between lagged versions of a time-series, and can be used to identify the presence of cyclic patterns in the data. In the context of long-term linear prediction, autocorrelation can be used to identify the lag that maximizes the correlation between past and future values of the time-series. This lag can be used as the optimal shift for the long-term linear prediction model. To use autocorrelation for detecting the optimal shift in long-term prediction, one would first calculate the autocorrelation function (ACF) for the time-series data. The ACF measures the correlation between the time-series at different lags. The lag at which the ACF is highest corresponds to the optimal shift for long-term linear prediction. Once the optimal shift has been identified, it can be used to train a long-term linear prediction model using techniques such as autoregressive models, moving average

models, or combinations of both. However, it's important to note that while autocorrelation can be a useful tool for identifying the optimal shift for long-term linear prediction, it may not always be the most appropriate technique for all types of time-series data. Other methods, such as spectral analysis or wavelet analysis, may be more appropriate in some cases.

Autocorrelation function (ACF) of a time series is typically denoted as ρ_k or r_k and can be expressed mathematically using the following equation:

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2} \tag{1.4}$$

where y_t is the value of the time series at time t, $\bar{y}$ is the mean of the time series, k is the lag or time shift, γ_k is the autocovariance at lag k, γ_0 is the autocovariance at lag 0, and T is the total number of observations in the time series. The ACF measures the correlation between the time series and a lagged version of itself, with values ranging between -1 and 1.

1.4.5 Shift in long-term linear prediction

The shift of long-term linear prediction in human speech refers to the way in which the human vocal system produces sounds over time. Specifically, it describes the changes that occur in the speech signal over relatively long periods of time, typically measured in tens to hundreds of milliseconds. Long-term linear prediction is a technique used in speech analysis and synthesis to model the way that the vocal system produces speech sounds. It involves breaking the speech signal into short segments and using mathematical algorithms to predict the signal in each segment based on the signal in preceding segments. The shift of long-term linear prediction in human speech refers to the fact that the vocal system is constantly adjusting and adapting the way that it produces sounds based on a variety of factors, including the phonemes being spoken, the speaker's emotions and intentions, and the context of the speech. This means that the speech signal is not static, but rather is constantly shifting and evolving over time. For example, when a speaker is emphasizing a particular word or phrase, they may change the pitch, intensity, or duration of certain sounds in order to convey their meaning more effectively. These changes will be reflected in the long-term linear prediction of the speech signal, which will show shifts in the predicted signal over time.

In the context of stock data analysis, the shift in long-term prediction refers to

the way in which the patterns in the data change over time, reflecting changes in the underlying market conditions and investor sentiment. Long-term linear prediction is a technique that can be used to model and forecast stock prices based on historical price data. It involves breaking the time-series data into short segments and using mathematical algorithms to predict the price in each segment based on the price in preceding segments. The shift in long-term prediction for stock data can occur due to a variety of factors such as changes in the macroeconomic environment, company earnings announcements, news events, investor sentiment, and other market-moving events. These shifts can cause changes in the underlying trends and patterns in the data, which can make it difficult to accurately predict stock prices over longer time horizons. For example, sudden changes in market conditions such as a global financial crisis or a political event can cause significant shifts in long-term prediction for stock data, making it difficult to accurately forecast future prices. Similarly, a company's financial performance or regulatory changes can cause sudden shifts in the long-term prediction for that company's stock. Overall, the shift in long-term prediction for stock data reflects the dynamic and unpredictable nature of the stock market, and underscores the importance of using multiple sources of data and analysis techniques to make informed investment decisions.

To calculate the shift for long-term linear prediction, you can use the autocorrelation function of the signal. The shift value corresponds to the lag at which the autocorrelation function has its maximum value.

1.5 Detection of repeatable patterns

To detect repeatable patterns in time series data, you can use several techniques. Here are a few methods that can be used:

1. Autocorrelation: Autocorrelation measures the correlation between the time series data at different time lags. By calculating the autocorrelation at different lags, you can identify whether there are any repeating patterns in the data. If there are repeating patterns, the autocorrelation plot will show peaks at the lags corresponding to the pattern.

2. Seasonal decomposition: Seasonal decomposition separates the time series data into different components such as trend, seasonal, and residual. By analyzing the seasonal component, you can identify whether there are any

repeating patterns in the data. If there are repeating patterns, the seasonal component will show a consistent pattern over time.

3. Fourier transform: Fourier transform can be used to analyze the frequency components of the time series data. By applying Fourier transform, you can identify whether there are any repeating frequency patterns in the data.

4. Wavelet analysis: Wavelet analysis is a mathematical technique that can be used to analyze the time-frequency structure of the time series data. By applying wavelet analysis, you can identify whether there are any repeating patterns at different frequency scales.

5. Hidden Markov models: Hidden Markov models can be used to identify repeating patterns in the data. These models assume that the data follows a probabilistic process, where the state of the process is not directly observable. By estimating the model parameters, you can identify the underlying pattern and predict future values of the time series data.

6. PDMD alghoritm developed on University of California, Riverside

These techniques can be used individually or in combination to detect repeatable patterns in time series data. It's important to note that the choice of technique depends on the characteristics of the data and the specific problem being addressed.

1.5.1 Hidden Markov models

Hidden Markov Models (HMMs) are probabilistic models that are used to analyze sequential data where the state of the system is not directly observable. HMMs consist of a set of hidden states that are not directly observable, and a set of observable symbols that are emitted from the hidden states.

The HMM assumes that the hidden states form a Markov chain, which means that the probability of transitioning from one hidden state to another depends only on the current state and not on the previous states. The emission probabilities of the observable symbols depend only on the current hidden state.

HMMs are typically used for two main tasks:

1. Evaluation: Given a sequence of observable symbols, HMMs can be used to evaluate the probability of the sequence of symbols occurring, given a particular set of model parameters.

2. Decoding: Given a sequence of observable symbols, HMMs can be used to determine the most likely sequence of hidden states that generated the sequence of symbols.

3. HMMs are used in a wide range of applications such as speech recognition, handwriting recognition, natural language processing, bioinformatics, and financial time series analysis.

The basic steps involved in building an HMM model are as follows:

1. Define the set of hidden states: Determine the set of hidden states that the system can be in at any given time.

2. Define the set of observable symbols: Determine the set of observable symbols that can be emitted from each hidden state.

3. Define the transition probabilities: Determine the probability of transitioning from one hidden state to another.

4. Define the emission probabilities: Determine the probability of emitting each observable symbol from each hidden state.

5. Estimate the model parameters: Estimate the transition and emission probabilities of the HMM using a training dataset.

6. Use the model for evaluation or decoding: Once the model parameters have been estimated, the HMM can be used for evaluation or decoding tasks.

It's important to note that building an HMM model requires knowledge of probability theory and statistical modeling, and may require significant computational resources.

1.5.2 Fourier transform

Fourier transform is a mathematical technique that is used to analyze the frequency components of a signal or a time series data. It transforms a signal from the time domain into the frequency domain, which allows us to analyze the signal in terms of its frequency components. The Fourier transform is widely used in many fields, including signal processing, image processing, and physics.

The Fourier transform decomposes a signal into a set of sinusoidal waves with different frequencies, amplitudes, and phases. These sinusoidal waves are called

the Fourier series, and they represent the signal as a sum of complex exponential functions.

The Fourier transform can be expressed as an integral equation that takes a time-domain signal and produces a frequency-domain representation of that signal. The formula for the Fourier transform is:

$$F(\omega) = \int f(t)e^{(-i\omega t)}dt, \tag{1.5}$$

where $F(\omega)$ is the frequency-domain representation of the signal, $f(t)$ is the time-domain signal, ω is the frequency, and i is the imaginary unit.

The inverse Fourier transform can be used to transform the frequency-domain representation of a signal back into the time domain. The formula for the inverse Fourier transform is:

$$f(t) = (\frac{1}{2\pi}) \int F(\omega)e^{(i\omega t)}d\omega \tag{1.6}$$

where $f(t)$ is the time-domain signal, $F(\omega)$ is the frequency-domain representation of the signal, ω is the frequency, and i is the imaginary unit.

The Fourier transform has many practical applications. For example, it can be used to analyze the frequency components of a music signal, to filter out noise from a signal, or to compress data by removing high-frequency components that are not essential for the representation of the signal. It is a powerful tool for understanding the underlying structure of signals and is widely used in scientific research and engineering applications.

1.5.3 Seasonal decomposition

Seasonal decomposition is a statistical technique that is used to decompose a time series into its underlying components, including trend, seasonal, and residual components. This technique is useful for identifying and understanding the repeating patterns or seasonal effects in a time series data.

The seasonal decomposition of a time series involves separating the data into four components:

1. Trend component: This component represents the long-term pattern in the data, such as increasing or decreasing trends over time.

2. Seasonal component: This component represents the repeating pattern in the data that occurs over a fixed period, such as daily, weekly, or monthly patterns.

3. Residual component: This component represents the remaining variation in the data that cannot be explained y the trend or seasonal components. It may include random noise or other unexplained factors.

4. Irregular component: This component represents any unexpected or irregular variation in the data that is not accounted for by the other components.

The seasonal decomposition process involves applying a smoothing algorithm to the time series data to estimate the trend and seasonal components, and then subtracting them from the original data to obtain the residual component. There are various methods for performing seasonal decomposition, including moving averages, exponential smoothing, and regression models.

Once the components of the time series have been separated, they can be analyzed and modeled separately. This can help in detecting the repeating patterns or seasonal effects in the data and in making accurate forecasts for future time periods.

Overall, seasonal decomposition is a powerful tool for identifying and analyzing the seasonal patterns in a time series data, and it can be useful in a wide range of applications, including finance, economics, and meteorology.

1.5.4 PDMD algorithm

PDMD (Probabilistic Data Mining on Data Streams) is an algorithm for online learning and prediction in data streams that was developed by KDD Lab at University of California, Riverside. It is a probabilistic model-based approach that can learn from a stream of input data and make predictions in real-time.
PDMD uses a probabilistic generative model to represent the joint distribution of the input data and the target variable. The model is updated in an online manner as new data arrives, using a Bayesian inference method to estimate the posterior distribution of the model parameters. This allows PDMD to adapt to changes in the data distribution over time.
To make predictions, PDMD calculates the posterior predictive distribution of the

target variable given the observed data. This distribution represents the uncertainty in the prediction, and can be used to estimate the expected value of the target variable or to calculate prediction intervals.
PDMD has been applied to various applications such as anomaly detection, classification, and regression on data streams. It has shown to be effective in handling data streams with changing distributions, noisy data, and missing values.
Overall, PDMD is a promising algorithm for online learning and prediction in data streams that can adapt to changes in the data distribution over time and provide probabilistic predictions.

2 Goal of the thesis

In this thesis we are focus on improving linear prediction in sales and financial forecasting and combine it with modern machine learning approaches. Machine learning techniques are used to improve the accuracy of linear prediction models. Specifically, there are two ways in which machine learning are applied to linear prediction:

1. Feature engineering: In this approach, machine learning is used to extract relevant features from the input signal that can be used as input to a linear prediction model.The extracted features can capture complex patterns in the data that are not captured by the raw input. Feature engineering can be done using techniques such as principal component analysis, wavelet transform, and Fourier transform.

2. Model selection and training: In this approach, machine learning is used to select the best linear model for the prediction task and to estimate its parameters from the data. This can involve selecting the best set of input variables for the linear model, choosing the best regularization parameter to avoid overfitting, and optimizing the model hyperparameters. Common machine learning algorithms used for linear prediction include linear regression, support vector regression, and artificial neural networks.

The goal of long-term linear prediction is to estimate future values of a signal or time series based on its past values using a linear model. The linear prediction model uses a set of coefficients to weight past values of the signal and produce a prediction for future values. The accuracy of the prediction depends on the quality of the model and the complexity of the underlying signal. Combine this principles get to us the best results from linear prediction over sales and financial datasets.

3 Methodology

3.1 Characteristics of the research object

Sales data and financial datasets are two broad categories of data that have different characteristics and are used for different purposes. Here are some general characteristics of each type of dataset:

Sales data
Typically contains transactional information, such as the date, time, location, and amount of a purchase Can include additional information about the customer, such as their demographic profile, purchase history, and preferences Often analyzed to understand customer behavior, such as buying patterns, trends, and preferences May have seasonal or cyclical patterns, depending on the nature of the product or service being sold Can be used to optimize marketing and sales strategies, such as targeting specific customer segments, promoting certain products, or adjusting prices and discounts.

Financial datasets
Typically contains financial information, such as the revenue, expenses, assets, liabilities, and cash flow of a company or organization Can include additional information about the market, such as interest rates, exchange rates, and stock prices Often analyzed to evaluate the financial performance and health of a company, such as profitability, solvency, and liquidity May have regulatory or compliance requirements, such as financial reporting standards or tax laws Can be used to make strategic decisions, such as investment, merger and acquisition, or divestiture Both sales data and financial datasets can be used for forecasting and modeling, but they have different analytical techniques and tools. Sales data often requires customer segmentation, predictive analytics, and machine learning algorithms, while financial datasets require financial ratio analysis, time series forecasting, and risk assessment.

3.2 Methods

3.2.1 Linear regression

Linear regression [**linear**] attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, we want to relate the weights of individuals to their heights using a linear regression model. Before attempting to fit a linear model to observed data, we should first determine if there exists a relationship between the variables of interest. This does not necessarily mean that one variable causes the other, but that there is some significant association between them. To determine the strength of relationship a scatterplot can be a helpful tool. If there appears to be no association between the proposed explanatory and dependent variables, then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation coefficient. That is a value from $[-1, 1]$ range indicating the strength of the association of the observed data for the two variables.
A linear regression line has an equation of the form $Y = a + bX$, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b, and a is the intercept. Example of some basic linear models:

$$\begin{aligned} Y &= ax + b \\ Y &= a + bx + c \\ Y &= a \sin x + b \end{aligned} \tag{3.1}$$

3.2.2 Linear prediction

Linear prediction is a statistical technique used to forecast future values based on past observations. It is a method for modeling the relationship between a dependent variable and one or more independent variables in a linear form. The goal of linear prediction is to find the best linear approximation of the relationship between the variables, which can then be used to make predictions about future values of the dependent variable.

3.2.3 Backpropagation

Backpropagation is a widely used algorithm for computing the gradients of the loss function with respect to the weights of a neural network. It is the backbone

of training deep neural networks using stochastic gradient descent (SGD) or its variants. The backpropagation algorithm computes the gradient of the loss function with respect to each weight in the network by recursively applying the chain rule of differentiation. The algorithm is typically implemented in two phases:

1. Forward pass: The forward pass involves computing the output of each layer in the network, starting from the input layer and propagating through the hidden layers to the output layer. The output of each layer is computed as a function of the input to the layer and the weights of the layer. The forward pass computes the predictions of the network on a given input.

2. Backward pass: The backward pass involves computing the gradients of the loss function with respect to each weight in the network, starting from the output layer and propagating backwards through the hidden layers to the input layer. The gradients are computed by applying the chain rule of differentiation to the output of each layer. The gradients are then used to update the weights of the network using an optimization algorithm such as SGD.

The backpropagation algorithm can be optimized using various techniques, such as parallel computing, weight sharing, and regularization. It is a powerful tool for training deep neural networks with many layers and millions of parameters, and has enabled significant advances in many areas of machine learning, including computer vision, natural language processing, and speech recognition.

3.3 Datasets

Creating a science research dataset involves several steps, which may vary depending on the nature of the research and the specific field of science. Here are some general steps to consider:

1. Determine the research question: The first step in creating a research dataset is to determine the research question. This will help you identify the data you need to collect and the type of dataset you need to create.

2. Choose the data sources: Once you have identified the research question, you need to determine the data sources you will use to create your dataset. Depending on the research question, you may use publicly available data, data from surveys or experiments, data from literature, or a combination of these.

3. Collect and organize the data: Collecting data can involve different methods, such as surveys, experiments, literature reviews, or data mining. You need to make sure that the data you collect is reliable, accurate, and relevant to your research question. Once you have collected the data, you need to organize it in a way that is easy to analyze.

4. Clean and preprocess the data: Before you can analyze the data, you need to clean and preprocess it. This involves removing any errors, missing values, or duplicates in the data. You may also need to transform or normalize the data to make it compatible with the analysis methods you plan to use.

5. Perform exploratory analysis: Once the data is cleaned and preprocessed, you can perform exploratory analysis to identify patterns, trends, or relationships in the data. This can help you refine your research question or identify areas that require further investigation.

6. Perform statistical analysis: Depending on the research question, you may need to perform statistical analysis to test hypotheses or evaluate the significance of the results. This can involve using regression analysis, hypothesis testing, or other statistical methods.

Validate the dataset: After we have analyzed the data, we need to validate the dataset to ensure that the results are accurate and reliable. This can involve comparing the results to other datasets or conducting further experiments to verify the results. Overall, creating a science research dataset involves careful planning, data collection, preprocessing, and analysis to ensure that the data is accurate, reliable, and relevant to the research question.

3.4 Comparison criteria

Finally we define the method to compare our models results. Absolute number of income value prediction should not be important for the store owners because of that we calculated the aberration for each month prediction and then we easily calculate quarterly and yearly results. The sum of squared errors (SSE), defined by:

$$SSE = \sum_{i=1}^{n} w_i(y_i - \overline{y_i})^2,$$

between the fitting models and the used data serves as the fitting criterion, with values closer to 0 indicating a smaller random error component of the model.

Also some other quality measures were evaluated, *i.e.* the R-square from interval $[0,\ 1]$, that indicates the proportion of variance satisfactory explained by the fitting-model (*e.g.* R-square $= 0.7325$ means that the fit explains 73.25% of the total variation in the data about the average); R-square is defined as the ratio of the sum of squares of the regression (SSR) and the total sum of squares (SST). SSR is defined as

$$SSR = \sum_{i=1}^{n} w_i(\overline{y_i} - \overline{y_i})^2.$$

SST is also called the sum of squares about the mean, and is defined as

$$SST = \sum_{i=1}^{n} w_i(y_i - \overline{y})^2,$$

where SST = SSR + SSE. Givenm these definition, R-square is expressed as

$$\frac{SSR}{SST} = 1 - \frac{SSE}{SST}.$$

The adjusted R-square statistic, with values smaller or equal to 1, where values closer to 1 indicate a better fit; the root mean squared error (RMSE):

$$RMSE = s = \sqrt{\frac{SSE}{v}}$$

with values closer to 0 indicating a fit more useful for prediction.

Mean square error (MSE) is a commonly used metric to measure the average squared difference between the actual and predicted values of a continuous variable. It is a measure of how well a prediction model performs in predicting continuous outcomes.
To calculate the MSE, you first take the difference between the predicted and actual values for each data point, square these differences, and then take the average of the squared differences across all data points.

The formula for calculating MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{3.2}$$

where n is the total number of data points, y_i is the actual value of the $i-th$ data point, and $\hat{y}_i$ is the predicted value of the $i-th$ data point.

The square of the difference in each data point ensures that the errors are positive

and provides a way to measure the magnitude of the error. By squaring the errors, the MSE gives more weight to larger errors and less weight to smaller errors.

MSE is a popular metric used in many fields, such as machine learning, statistics, and engineering, to evaluate the performance of prediction models. A lower MSE indicates better performance of the model in predicting the outcomes.

3.5 Statistics methods

Median

The median is a statistical measure that represents the middle value of a dataset. It is a value that separates the dataset into two halves: half of the values are greater than the median, and half of the values are less than the median. In other words, the median is the value that is exactly in the middle of the dataset when the values are arranged in order of magnitude.

To compute the median of a dataset, we first sort the values in ascending or descending order. If the dataset has an odd number of values, the median is the middle value. For example, in the dataset $1, 2, 3, 4, 5$, the median is 3. If the dataset has an even number of values, the median is the average of the two middle values. For example, in the dataset $1, 2, 3, 4$, the median is $(2 + 3)/2 = 2.5$.

The median is a robust statistic, meaning that it is not affected by outliers or extreme values in the dataset, unlike the mean. This makes it a useful measure of central tendency in datasets with a large number of outliers or skewed distributions. The median is commonly used in various applications, such as finance, economics, and social sciences, to summarize and compare datasets.

Standard deviation

Standard deviation is a statistical measure that quantifies the amount of variation or dispersion in a dataset. It measures how far the values in a dataset deviate from the mean, or average, of the dataset. The standard deviation is a non-negative number and has the same units as the data being measured.

To compute the standard deviation of a dataset, we first calculate the mean of the dataset. Then, we calculate the difference between each value in the dataset and the mean, square each difference, and sum up the squared differences. Finally, we divide the sum of squared differences by the number of values in the dataset, and take the square root of the result. This gives us the standard deviation of the dataset.

A small standard deviation indicates that the values in the dataset are tightly clustered around the mean, while a large standard deviation indicates that the values are widely spread out from the mean. Standard deviation is commonly used in various applications, such as finance, engineering, and natural sciences, to analyze and compare datasets. It is also an important parameter in many statistical tests and models, such as the normal distribution and the t-test.

4 Syntactic part

Based on the Analytical part 1 let us create new mathematical models and approaches to made a fast and accuracy sales forcasting consist of long-therm linear prediction with individual weights calculated for each period all based on Levinson-Durbin scheme caled Extended linear prediction (ELP) We expect to get better results than by using prediction based on short-term or long-term standard linear prediction (see section 1.4). Finally, our approach will return future values for sales companies based on previous data with better aberration than linear prediction has.

4.1 Order calculation

Create neural site for order prediction

Creating a neural network for linear prediction and optimal order detection involves several steps. Here's a general overview of the process:

1. Data Preparation: Collect a dataset of input/output pairs that represent the relationship you want the neural network to learn. For linear prediction, this could be a time-series dataset where you want to predict the next value based on the previous values. For optimal order detection, this could be a dataset where you have inputs and the corresponding optimal order.

2. Data Preprocessing: Preprocess the dataset by normalizing the inputs and outputs to a common range, shuffling the dataset, and splitting it into training, validation, and testing sets.

3. Model Architecture: Choose an appropriate neural network architecture for your problem. For linear prediction, you could use a recurrent neural network (RNN) such as LSTM or GRU. For optimal order detection, you could use a feedforward neural network.

4. Training: Train the neural network on the training set using an appropriate optimization algorithm such as stochastic gradient descent (SGD), Adam, or RMSProp. During training, monitor the performance of the network on the validation set to detect overfitting and adjust the hyperparameters accordingly.

5. Testing: Evaluate the performance of the trained neural network on the testing set to get an estimate of its generalization ability.

6. Deployment: Deploy the trained neural network to make predictions on new data.

Choose activation function

Choosing the right activation function for linear prediction order detection in neural networks depends on several factors such as the type of data, the complexity of the problem, and the architecture of the neural network. Here are some guidelines to help you choose the right activation function:

1. Linear activation function: If you want to perform linear regression or linear prediction, you can use a linear activation function. The output of a linear activation function is proportional to the input, which makes it ideal for linear regression problems.

2. ReLU activation function: The Rectified Linear Unit (ReLU) activation function is one of the most popular activation functions used in deep learning. ReLU is simple to compute and provides good results for most classification and regression problems.

3. Sigmoid activation function: The Sigmoid activation function maps any input value to a value between 0 and 1. It is commonly used in binary classification problems, where the goal is to classify data into two categories.

4. Tanh activation function: The hyperbolic tangent (Tanh) activation function is similar to the sigmoid function but maps input values to a range between -1 and 1. It is also used in binary classification problems.

5. Softmax activation function: The Softmax activation function is used in multi-class classification problems. It maps input values to a probability distribution over the output classes.

6. In general, it's a good idea to start with the ReLU activation function and see if it works well for your problem. If not, you can try other activation functions that are appropriate for your problem. It's also important to experiment with different architectures and hyperparameters to find the best model for your data.

Train neural site

Training a neural network involves several steps:

1. Preparing the data: This involves collecting and preprocessing the data that will be used to train the neural network. The data needs to be cleaned, normalized, and split into training, validation, and testing sets.

2. Defining the architecture: This involves selecting the number and type of layers, the activation functions, the optimization algorithm, and the loss function. The architecture should be chosen based on the problem being solved.

3. Initializing the weights: The weights of the neural network are initialized randomly.

4. Forward propagation: The input data is fed into the neural network, and the outputs of each layer are computed using the activation functions.

5. Backward propagation: This involves computing the gradient of the loss function with respect to the weights of the neural network. The gradients are computed using the chain rule of differentiation.

6. Updating the weights: The weights of the neural network are updated using an optimization algorithm such as stochastic gradient descent.

7. Testing the model: Once the model is trained, it needs to be tested on a separate dataset to evaluate its performance.

8. Fine-tuning the model: Based on the performance of the model, adjustments can be made to the architecture, hyperparameters, or training data to improve its performance.

9. The above steps are repeated multiple times until the performance of the model reaches a satisfactory level. It's important to note that training a neural network is a computationally intensive task, and it may take a significant amount of time and resources.

Run neural site to get optimal order

Once a neural network is trained, there are several steps you can take to get the best performance out of it:

1. Use the validation set to tune hyperparameters: The validation set is a dataset that is separate from the training and testing sets and is used to tune hyperparameters such as learning rate, batch size, and number of epochs. By experimenting with different hyperparameters, you can find the optimal settings that result in the best performance.
2. Test the model on a separate dataset: Once the model is trained and the hyperparameters are tuned, it's important to test the model on a separate dataset to evaluate its performance. This dataset should be completely separate from the training and validation sets.
3. Use data augmentation techniques: Data augmentation techniques such as rotation, translation, and scaling can be used to generate additional training data, which can improve the performance of the model.
4. Use ensembling techniques: Ensembling techniques such as bagging and boosting can be used to combine the predictions of multiple models to improve the overall performance.
5. Use regularization techniques: Regularization techniques such as L1 and L2 regularization can be used to prevent overfitting and improve the generalization of the model.
6. Fine-tune the model on new data: If new data becomes available, the model can be fine-tuned on the new data to further improve its performance.
7. By following these steps, you can get the best performance out of a trained neural network. It's important to remember that there is no single approach that works best for all problems, and experimentation is often necessary to find the optimal solution.

4.2 Shift calculation

4.2.1 Autocorrelation

1. Compute the autocorrelation function of the signal using the xcorr function in MATLAB. The xcorr function returns the cross-correlation sequence of

the signal with itself.

$$[Rxx, lag] = xcorr(x, maxlag);$$

where x is the input signal and maxlag is the maximum lag to compute. Rxx is the cross-correlation sequence, and lag is the corresponding lag vector.

2. Find the index of the maximum value in the autocorrelation function. You can use the max function to find the maximum value and its index.

$$[\ , idx] = max(Rxx);$$

The ignores the value of the maximum, and idx is the index of the maximum value in the Rxx vector.

3. Calculate the shift value from the lag index. The shift value is simply the lag value corresponding to the maximum value in the autocorrelation function.

$$shift = lag(idx);$$

The shift value is the lag at which the autocorrelation function has its maximum value, which corresponds to the shift value for long-term linear prediction. Note that the shift value is expressed in samples, so we may need to convert it to a time delay if your signal has a specific sampling rate.

4.2.2 Neural network

Create neural site for shift and long shift prediction

To create a neural network for shift and long shift prediction, you can follow these steps:

1. Define the problem: The first step is to define the problem you want to solve. In this case, the problem is to predict the shift and long shift of a time series data.

2. Collect and preprocess the data: Collect the data that you want to use for training and testing the neural network. Preprocess the data by normalizing it and splitting it into training, validation, and testing sets.

3. Choose the architecture: Choose the architecture for your neural network based on the problem you want to solve. In this case, a recurrent neural network (RNN) architecture such as a Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) may be suitable, as they are specifically designed to handle sequential data.

4. Define the input and output: Define the input and output of your neural network. The input should be the time series data, and the output should be the predicted shift and long shift.

5. Define the loss function: Choose an appropriate loss function to measure the difference between the predicted shift and long shift and the actual shift and long shift.

6. Train the model: Train the model using the training data and tune the hyperparameters to improve the performance of the model. Monitor the model's performance using the validation set.

7. Evaluate the model: Evaluate the performance of the model on the testing set to see how well it generalizes to new data.

8. Fine-tune the model: If the model does not perform well on the testing set, fine-tune the model by adjusting the architecture, hyperparameters, or training data.

9. Use the model: Once you have a well-performing model, you can use it to predict the shift and long shift of new time series data.

By following these steps, you can create a neural network for shift and long shift prediction. It's important to experiment with different architectures and hyperparameters to find the best model for your data.

Choose activation function

Choosing the activation function for a neural network for shift and long shift prediction depends on the architecture of the network and the problem being solved.

For a recurrent neural network (RNN) architecture such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), the most commonly used activation function is the hyperbolic tangent (tanh) function. This is because the tanh function produces outputs in the range [-1, 1], which makes it suitable for RNNs that

use recurrent connections to store and update memory over time.

However, other activation functions such as the Rectified Linear Unit (ReLU) or its variants, such as leaky ReLU or exponential ReLU, can also be used in RNNs for shift and long shift prediction. These activation functions are computationally efficient and can be used in deep neural networks without the problem of vanishing gradients.

In general, the choice of activation function should be based on the characteristics of the problem being solved, the architecture of the network, and the properties of the activation function itself. Experimentation with different activation functions can help to determine the most appropriate one for a given problem.

Train neural site

To train a neural network for shift and long shift prediction, you can follow these steps:

1. Collect and preprocess the data: Collect the time series data that you want to use for training and testing the neural network. Preprocess the data by normalizing it and splitting it into training, validation, and testing sets.

2. Define the architecture: Choose the architecture for your neural network based on the problem you want to solve. For shift and long shift prediction, a recurrent neural network (RNN) architecture such as a Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) may be suitable.

3. Define the input and output: Define the input and output of your neural network. The input should be a sequence of data points, and the output should be the predicted shift and long shift.

4. Define the loss function: Choose an appropriate loss function to measure the difference between the predicted shift and long shift and the actual shift and long shift. Mean Squared Error (MSE) is commonly used for regression problems such as this.

5. Train the model: Train the model using the training data and tune the hyperparameters to improve the performance of the model. Use the validation set to monitor the model's performance and avoid overfitting.

6. Evaluate the model: Evaluate the performance of the model on the testing set to see how well it generalizes to new data. Compare the predicted shift and long shift values with the actual values.

7. Fine-tune the model: If the model does not perform well on the testing set, fine-tune the model by adjusting the architecture, hyperparameters, or training data.

8. Use the model: Once you have a well-performing model, you can use it to predict the shift and long shift of new time series data.

It's important to experiment with different architectures and hyperparameters to find the best model for your data. Additionally, it's important to properly preprocess the data and monitor the model's performance to avoid overfitting. With these steps, you can train a neural network for shift and long shift prediction.

Run neural site to get optimal shift and long shift

To run a trained neural network for shift and long shift prediction, you can follow these steps:

1. Preprocess the data: If you have new data that you want to use for prediction, preprocess it in the same way as the training data by normalizing it and formatting it into sequences.

2. Load the trained model: Load the trained model into memory using a deep learning framework such as Tensorflow, PyTorch, or Keras.

3. Prepare the input data: Prepare the input data by formatting it into sequences that match the input shape of the model.

4. Make predictions: Use the predict function of the model to make predictions on the input data. The output of the model will be the predicted shift and long shift values.

5. Postprocess the output: If necessary, postprocess the output of the model by denormalizing it or transforming it back into the original format.

6. Evaluate the predictions: Evaluate the quality of the predicted shift and long shift values by comparing them with the actual shift and long shift values. Calculate performance metrics such as mean squared error or mean absolute error to quantify the accuracy of the predictions.

7. Refine the model: If the predicted shift and long shift values are not accurate enough, refine the model by adjusting the architecture, hyperparameters, or training data and repeat the process.

With these steps, you can run a trained neural network for shift and long shift prediction and obtain optimal shift and long shift values. It's important to note that the quality of the predictions depends on the quality and quantity of the data used for training and the architecture and hyperparameters of the model. Therefore, it's important to experiment with different models and training data to find the optimal solution.

4.3 Weights for each period

To create a mathematical model for predicting sales data with periodical trends, you can use a seasonal ARIMA (SARIMA) model. This model takes into account seasonal variations in the data and uses autoregressive and moving average terms to capture the patterns and trends in the data.

To create the weights for the SARIMA model, you can follow these steps:

1. Identify the seasonal period: Determine the seasonal period of the sales data. This can be done by analyzing the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the data.

2. Estimate the parameters: Estimate the parameters of the SARIMA model using the sales data. This can be done using statistical software such as R or Python.

3. Interpret the weights: Once the parameters have been estimated, interpret the weights of the model to understand how they contribute to the prediction. The weights represent the strength of the relationship between the sales data and the predictor variables.

4. Validate the model: Validate the model by comparing the predicted sales data with the actual sales data. Use statistical measures such as mean squared error or mean absolute error to evaluate the accuracy of the predictions.

5. Refine the model: Refine the model by adjusting the weights or the parameters of the model based on the validation results. Repeat the validation process until the model provides accurate predictions.

It's important to note that creating a SARIMA model can be complex and requires knowledge of time series analysis and statistical modeling. Therefore, it's recommended to seek the help of a data scientist or a statistician for assistance.

4.4 Extended long-term prediction

4.5 Combining all principles to forecast process

1. Run neural site to get right order of linear prediction
2. Run neural site to get righ shift and long shift of linear prediction
3. Run extended long-term prediction to ger predicted values of sales data
4. Apply periodical wages to previous response data
5. Calculate comparison criteria for results and plot data and mean square errors

5 Evaluation

5.1 Experiment

5.1.1 Preprocessing of input data

5.1.2 Models for sales forecasting

5.1.3 Results

5.2 Matlab Live script application

6 Summary

Bibliography

1. VAIDYANATHAN, P. p. *The Theory of Linear Prediction*. 2007.
2. MATHWORKS. *What Is a Live Script or Function?* 2021. `https://www.mathworks.com/help/matlab/matlab_prog/what-is-a-live-script-or-function.html`.
3. CRYER, Jonathan D. *Time series analysis*. Duxbury Press Boston, 1986.
4. FAHRMEIR, Ludwig; KNEIB, Thomas; LANG, Stefan; MARX, Brian D. Regression models. In: *Regression: Models, methods and applications*. Springer, 2022, pp. 23–84.
5. KOTSIANTIS, Sotiris B. Decision trees: a recent overview. *Artificial Intelligence Review*. 2013, vol. 39, pp. 261–283.
6. CHEN, Po-Hsuan Cameron; LIU, Yun; PENG, Lily. How to develop machine learning models for healthcare. *Nature materials*. 2019, vol. 18, no. 5, pp. 410–414.
7. MOURGIAS-ALEXANDRIS, G.; TSAKYRIDIS, A.; PASSALIS, N.; TEFAS, A.; VYRSOKINOS, K.; PLEROS, N. An all-optical neuron with sigmoid activation function. *Opt. Express*. 2019, vol. 27, no. 7, pp. 9620–9630. Available from DOI: `10.1364/OE.27.009620`.
8. GU, Jiuxiang; WANG, Zhenhua; KUEN, Jason; MA, Lianyang; SHAHROUDY, Amir; SHUAI, Bing; LIU, Ting; WANG, Xingxing; WANG, Gang; CAI, Jianfei; CHEN, Tsuhan. Recent advances in convolutional neural networks. *Pattern Recognition*. 2018, vol. 77, pp. 354–377. ISSN 0031-3203. Available from DOI: `https://doi.org/10.1016/j.patcog.2017.10.013`.
9. MEDSKER, Larry R; JAIN, LC. Recurrent neural networks. *Design and Applications*. 2001, vol. 5, pp. 64–67.
10. CHENG, Jianpeng; DONG, Li; LAPATA, Mirella. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*. 2016.

11. LUO, Tie; NAGARAJAN, Sai G. Distributed anomaly detection using autoencoder neural networks in WSN for IoT. In: *2018 ieee international conference on communications (icc)*. 2018, pp. 1–6.

12. CRESWELL, Antonia; WHITE, Tom; DUMOULIN, Vincent; ARULKUMARAN, Kai; SENGUPTA, Biswa; BHARATH, Anil A. Generative adversarial networks: An overview. *IEEE signal processing magazine*. 2018, vol. 35, no. 1, pp. 53–65.

13. FÉRAUD, Raphael; CLÉROT, Fabrice. A methodology to explain neural network classification. *Neural networks*. 2002, vol. 15, no. 2, pp. 237–246.

14. PARKS, P. Further comments on "An symmetric matrix formulation of the Hurwitz-Routh stability criterion. *IEEE Transactions on Automatic Control*. 1963, vol. 8, no. 3, pp. 270–271.

15. PHAM, Dinh Tuan; LE BRETON, Alain. Levinson-Durbin-type algorithms for continuous-time autoregressive models and applications. *Mathematics of Control, Signals and Systems*. 1991.

16. DURBIN, J. *The Fitting of Time-Series Models*. 1960.

17. RIAHY, G.H.; ABEDI, M. Short term wind speed forecasting for wind turbine applications using linear prediction method. *Renewable Energy*. 2008, vol. 33, no. 1, pp. 35–41.

18. NAVE, G.; COHEN, A. ECG compression using long-term prediction. *IEEE Transactions on Biomedical Engineering*. 1993, vol. 40, no. 9, pp. 877–885. Available from DOI: 10.1109/10.245608.

19. D. N. BAKER R. L. McPherron, T. E. Cayton et all. Linear prediction filter analysis of relativistic electron properties at 6.6 RE. *Space Physics*. 1990, vol. 95, no. A9, pp. 15133–15140.

20. KINOSHITA, Keisuke; DELCROIX, Marc; NAKATANI, Tomohiro; MIYOSHI, Masato. Suppression of Late Reverberation Effect on Speech Signal Using Long-Term Multiple-step Linear Prediction. *IEEE Transactions on Audio, Speech, and Language Processing*. 2009, vol. 17, no. 4, pp. 534–545. Available from DOI: 10.1109/TASL.2008.2009015.

List of Appendixes

A Flowcharts

A.1 Short-term linear prediction

A.2 Long-term linear prediction

A.3 Extended long-term linear prediction

B Modeling different situations