Machine Intelligence Notes + Capstone Project

ANBEFALEDE FAGLIGE FORUDSÆTNINGER FOR AT DELTAGE I MODULET

Modulet bygger videre på viden opnået i modulerne algoritmer og datastrukturer, sandsynlighedsteori og lineær algebra

MODULETS INDHOLD, FORLØB OG PÆDAGOGIK

LÆRINGSMÅL

VIDEN

Den studerende skal opnå viden om følgende teorier og metoder:

- problemløsning vha. søgning og inferens
- modelbaseret beslutningstræfning
- inferens under usikkerhed
- læring fra erfaring og læring fra data

FÆRDIGHEDER

- anvende korrekt teknisk notation og terminologi i skrift såvel som tale
- anvende grundlæggende teknikker præsenteret i kurset til løsning af en konkret problemstilling
- gøre rede for centrale principper og algoritmer præsenteret i kurset

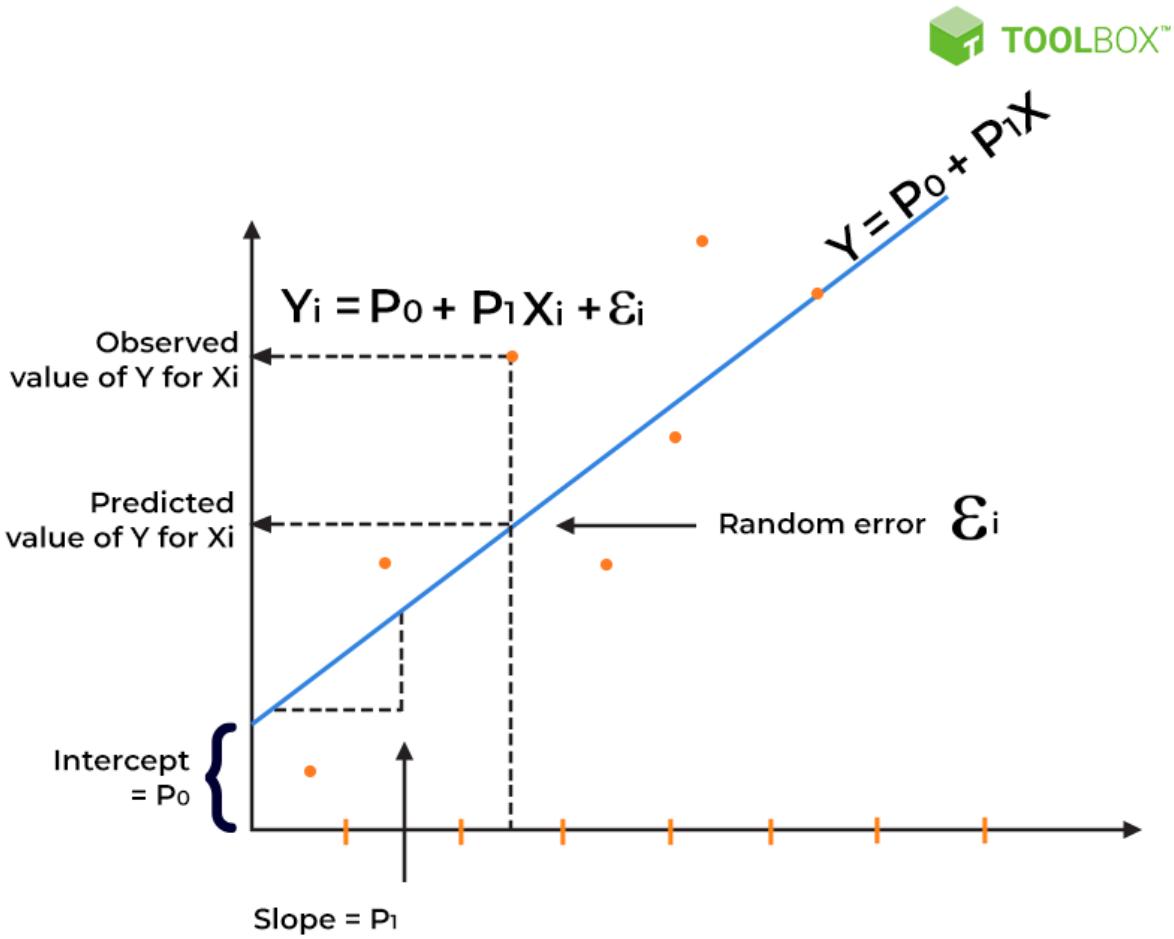
KOMPETENCER

- skal med udgangspunkt i en konkret problemstilling kunne vurdere, sammenligne og udvælge teknikker og metoder inden for maskinintelligens

Machine Intelligence Notes

Session 2 - Linear Regression I

Linear Regression Model



Definition:

- **Basic Idea:** Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables.
- **Formulation:** The model predicts the dependent variable (often denoted as y) as a linear combination of the independent variables (denoted as $x_1, x_2, ..., x_n$).

Mathematical Representation:

- **Simple Linear Regression:** $y = b_0 + b_1 x + \epsilon$
  - Where y is the dependent variable, x is the independent variable, b_0 is the y-intercept, b_1 is the slope of the line and ϵ is the error term.

- **Multiple Linear Regression:** $y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n + \epsilon$
    - Involves multiple independent variables influencing the dependent variables.

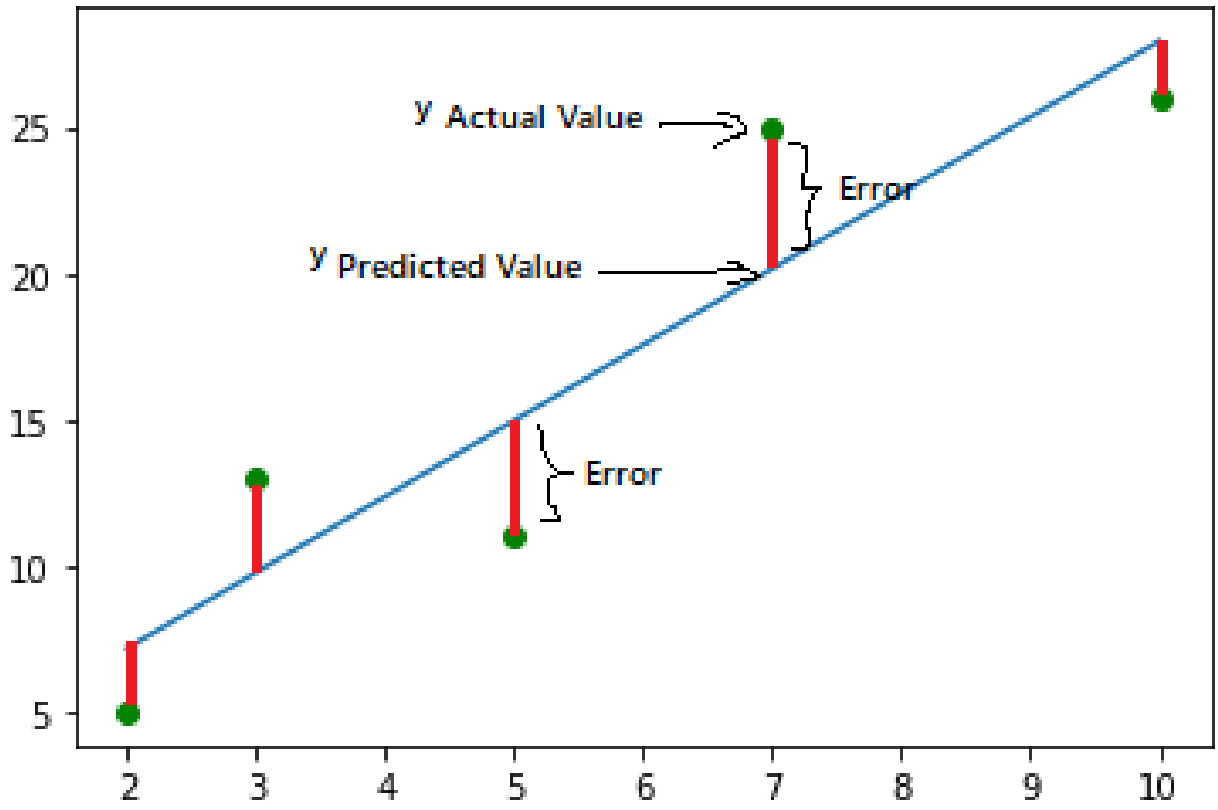
Assumptions:

- **Linearity:** The relationship between independent and dependent variables should be linear
- **Independence:** Observations should be independent of each other
- **Homoscedasticity**: The variance of residual is the same for any value of the independent variables.
- **Normal Distribution of Errors:** Residuals (difference between observed and predicted values) should be normally distributed.

Challenges and consideration:

- **Non-Linearity:** When the relationship between variables is not linear, linear regression may not be appropriate.
- **Outliers:** Extreme values can significantly affect the regression line
- **Multicollinearity:** In multiple linear regression, high correlation between independent variables can distort the model

Cost Functions in Linear Regression



Definition:

- **Purpose:** A cost function, also known as a loss function, measures how well the linear regression model is performing by calculating the difference between predicted values and the actual values.
- **Goal:** The objective of linear regression is to minimize the cost function, which translates to finding the best-fitting line through the data.

Common Types of Cost Functions in Linear Regression

- **Mean Squared Error (MSE):**
  - **Formula:** $MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$
  - **Explanation:** It calculates the average of the squares of the error, i.e., the average squared difference between the estimated values ($\hat{y}_i$) and the actual values (y_i).
  - **Usage:** Most common in linear regression models.
- **Mean Absolute Error (MAE):**

- **Formula:** $MAE = \frac{1}{n}\Sigma_{i=1}^{n}|y_i - \hat{y}_i|$
- **Explanation:** It measures the average magnitude of the errors in a set of predictions, without considering their direction
- **Usage:** Useful when you want to avoid sharing errors (as in MSE) which can disproportionately penalize large errors.

Properties

- **Convexity:** In linear regression, the cost function (like MSE) is convex, meaning it has a single global minimum
- **Differentiability:** These functions are typically smooth and differentiable, allowing the use of optimization algorithms like gradient descent.

Role in Model Training

- **Parameter Optimization:** By minimizing the cost function, the algorithm finds the most optimal parameters (coefficients) for the regression line.
- **Feedback Mechanism:** The cost function provides feedback to the model to adjust its weights.

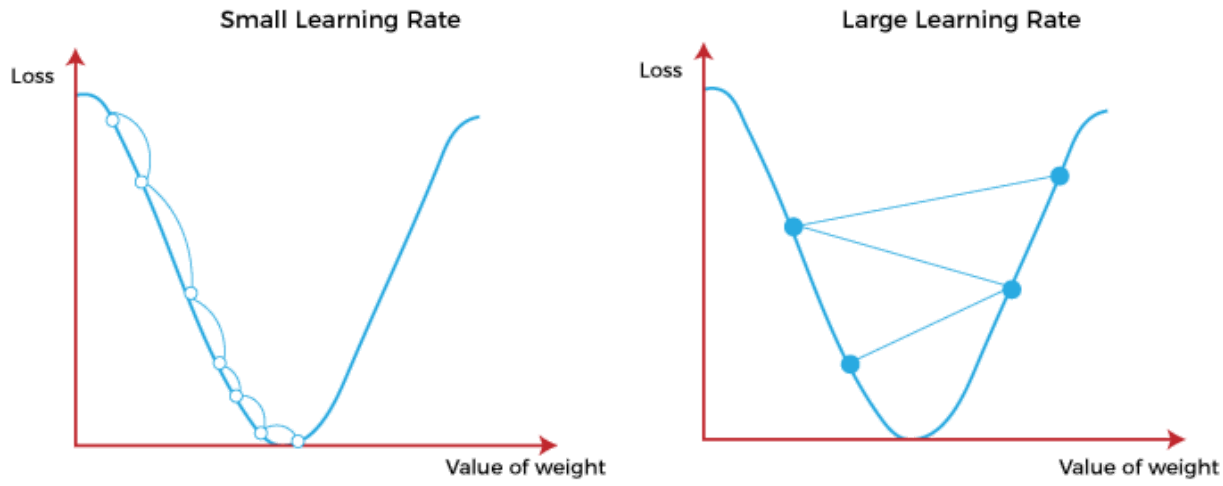
Challenges

- **Sensitive to Outliers:** MSE, for example, can be heavily influenced by outliers since it squares the errors.
- **Scaling Dependency:** The scale of the features can impact the function, making feature scaling an important preprocessing step, like normalizing feature values within a particular range (usually 0 to 1).

Importance in Machine Learning

- **Basis for Model Evaluation:** Cost functions are fundamental in evaluating the performance of a linear regression model.
- **Guide to Model Improvement:** By analyzing the cost function, one can understand where the model is lacking and what improvements are needed.

Gradient Descent in Linear Regression



Definition

- **Purpose:** Gradient Descent is an optimization algorithm used to minimize the cost function is a linear regression model. It is used to find the model parameters (coefficients) that result in the minimum cost.
- **Principle:** The algorithm iteratively adjusts the parameters to move towards the minimum value of the cost function.

How it Works

- **Initialization:** Start with initial values for the model parameters (often set to zero or a small random value).
- **Calculate Gradient:** Compute the gradient of the cost function with respect to each parameter. The gradient is a vector that points in the direction of the steepest increase of the function-
- **Update Parameters:** Adjust the parameters in the opposite direction of the gradient.
- **Iteration:** Repeat the process until the algorithm converges to the minimum value (i.e., the changes in the cost function are below a predefined threshold).

Formula

- **Parameter Update Rule:**

  $\theta = \theta - \alpha \times Gradient$

- Where θ represents the parameters, α is the learning rate (a small positive number), and $Gradient$ is the derivative of the cost function.

Key Components

1. **Learning Rate** (α)
   - Control how much we adjust the parameters with each step
   - Too small: The algorithm might take too long to converge.
   - Too large: might overshoot the minimum, leading to divergence.
2. **Convergence Criteria**
   - A rule to stop the iteration process is usually based on the change in the cost function between iterations.

Types of Gradient Descent

1. **Batch Gradient Descent:** Uses the entire training set to calculate the gradient at each step.
2. **Stochastic Gradient Descent (SGD):** Updates the parameters for each training example one by one.
3. **Mini-batch Gradient Descent:** Uses a subset of the training data to update parameters; balances efficiency and computational resource usage.

Challenges

- **Choosing the Right Learning Rate:** Requires experimentation and can significantly affect performance
- **Local Minima:** In more complex models (like neural networks), gradient descent can get stuck in local minima.
- **Computational Efficiency:** Especially in large datasets, the computation of gradients can be resource-intensive.

Session 3 - Linear Regression II

local minima—points where the cost function is lower than in the surrounding area but not the lowest possible value (global minimum).

Linear Regression with Multiple Features

Definition

Extends simple linear regression to accommodate multiple independent variables (features), enhancing the model's ability to explain the variance in the dependent variables.

Mathematical Formulation

- **Equation:** $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon$
    - Where y is the dependent variable, β_0 is the intercept, $\beta_1, \beta_2, ..., \beta_n$ are the coefficients for each feature, $x_1, x_2, ..., x_n$ are the independent variables and ϵ is the error term.

Key Aspects

1. **Multiple Independent Variables:**
    a. **Concept:** Involves more than one predictor variable influencing the dependent variable.
    b. **Challenge:** Increases complexity in understanding the relationship and interactions between different features.
2. **Hypothesis Function**
    a. **Definition:** Represents the predicted output of the model given the input features.
    b. **Multivariate Form:** Adjusted to include coefficients for each independent variable.
3. **Coefficients (β values):**
    a. **Role:** Determine the influence of each independent variable on the dependent variable.
    b. **Estimation:** Typically estimated using techniques like Gradient Descent
4. **Model Training**
    a. **Process:** Involves adjusting the model parameters to minimize a cost function, typically using techniques like Gradient Descent
    b. **Challenges:** More prone to overfitting due to increased complexity
5. **Model Evaluation**

a. **Metrics:** Utilizes R-squared, Adjusted R-squared, Mean Squared Error (MSE), etc., for assessing model performance.

b. **Consideration:** Adjusted R-squared is particularly useful as it accounts for the number of predictors in the model

6. **Interpretation of Coefficients**

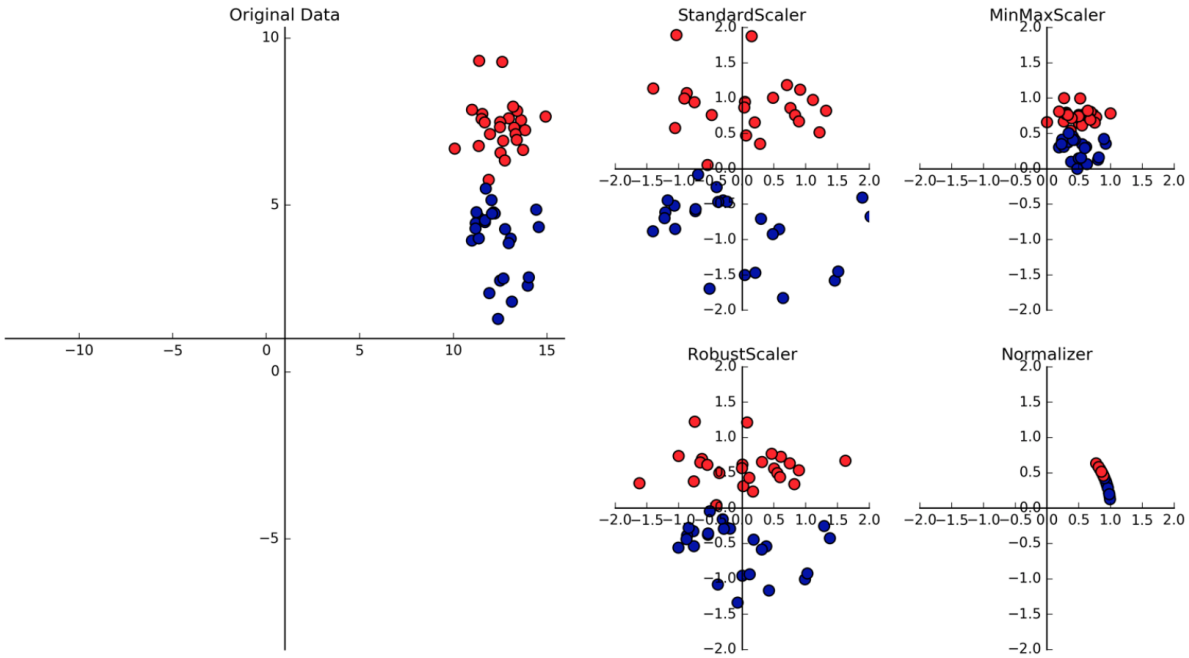
a. **Insight:** Provides an understanding of how changes in each independent variable are expected to influence the dependent variable.

b. **Cautions:** Must consider potential multicollinearity and ensure causal relationships are not assumed without further validation

Challenges and Considerations

- **Multicollinearity:** High correlation among independent variables can distort the model and make the interpretation of coefficients challenging.
- **Dimensionality:** Increased number of features can lead to higher computational complexity and risk of overfitting.

Feature Scaling/Normalization



Definition

Feature scaling or normalization is the process of standardizing the range of independent variables or features of data. In the context of linear regression, this ensures that each feature contributes equally to the determination of the output

Importance

- **Equal Influence:** Prevents features with larger numerical ranges from dominating those smaller ranges.
- **Convergence Speed:** Enhances the speed of convergence of gradient descent algorithms by ensuring all features are on a similar scale.

Common Methods

- **Min-Max Scaling (Normalization)**
  - **Formula:** $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$
  - **Result:** Transforms features to a scale from 0 to 1.
  - **Use Case:** When you want to bind features within a specific range.

bounds the features within a specific range, typically [0, 1]. This method is sensitive to outliers, which can skew the scaling.

- **Standardization (Z-score Normalization)**
  - **Formula:** $X_{standardized} = \frac{X - \mu}{\sigma}$
    - Where μ is the mean and σ is the standard deviation
  - **Results:** Transforms the features to have a mean of 0 and a standard deviation of 1
  - **Use Case:** Useful when data needs to be transformed into a standard normal distribution
- **Mean Normalization**
  - **Formula:** $X_{norm} = \frac{X - Average(X)}{Max(X) - Min(X)}$
  - **Result:** Similar to Min-Max scaling, but centers the data around zero.

Considerations

- **Choice of Method:** Depends on the dataset and model requirements. For instance, Z-score normalization is often preferred when the data distribution is known to be Gaussian

- **Application Consistency:** It's important to apply the same scaling method to both training and testing data.
- **Impact on Interpretation:** After scaling, the interpretation of coefficients in terms of the original scale of the data changes.

Handing Outliers

- **Sensitivity:** Min-Max scaling is sensitive to outliers, as it can compress most of the data into a small range. In contrast, Z-score normalization is less sensitive to outliers.

Applications in Machine Learning

- **Prerequisite for Many Models:** Essential for models that are sensitive to the scale of the data, like linear regression, k-nearest neighbors and neural networks.
  k-Nearest Neighbors (k-NN): This algorithm calculates distances between data points to make predictions.
- **Improving Model Performance:** Properly scaled features can significantly improve the performance and stability of machine learning models.

Challenges

- **Data Leakage:** Care must be taken to avoid data leakage when scaling, ensuring that the parameters used for scaling are derived from the training data alone

Feature Engineering

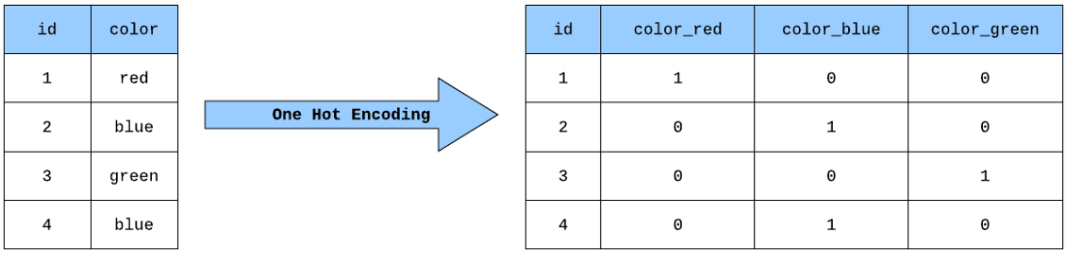
Definition

The process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data. These features can then be used to improve the performance of machine learning algorithms

Key Aspects

- **Selection of Relevant Features**
  - **Identifying Key Attributes:** Determining which features are more relevant to the prediction task.
  - **Removing Irrelevant or Redundant Data:** Helps in reducing overfitting and improving model performance

- **Creation of New Features**
  - **Combining Features**: Creating new features by combining two or more existing features.
  - **Transformation:** Applying transformations (like log, square, square root) to make the data linearly separable.
- **Handling Categorical Data**
  - **Encoding:** Transforming categorical variables into a form that an provided to ML algorithms (e.g., one-hot encoding)



- **Dealing with Missing Values**
  - **Imputation Techniques:** Filling in missing data with mean, median, mode or using more complex algorithms.

Challenges

- **Domain Knowledge:** Requires a good understanding of the domain to create meaningful features.
- **Dimensionality:** Adding too many features can increase the complexity of the model, leading to overfitting.
- **Balancing:** It's important to strike a balance between adding new features and the model's simplicity and interpretability.

Iterative Process

- **Continuous Refinement:** Feature engineering is often an iterative process, where features are continuously refined and evaluated based on model performance.

Overfitting

Overfitting is when a statistical model fits exactly against its training data. This means that the algorithm cannot perform accurately against unseen data. It should generalize the data and not memorize it, as generalization allows machine learning algorithms to run properly and make predictions and classify data. Essentially what happens is that the model trains too long on sample data and learns “noise” or irrelevant information within the dataset. If the model learns this noise and fits closely to the training set, then the model becomes “overfitted” and cannot generalize well to new data. If a model cannot generalize well to new data, then it will not be able to perform classification or prediction tasks properly. Indicators of overfitting are typically low error rates and a high variance. If the training data has a low error rate and the test data has a high error rate then it signals overfitting. Overfitting can be avoided by using for example data augmentation or regularization.

Session 4 - Logistic Regression

MEANING OVERFITTING:
model becomes too complex and captures the noise in the training data, which leads to poor generalization to unseen data.

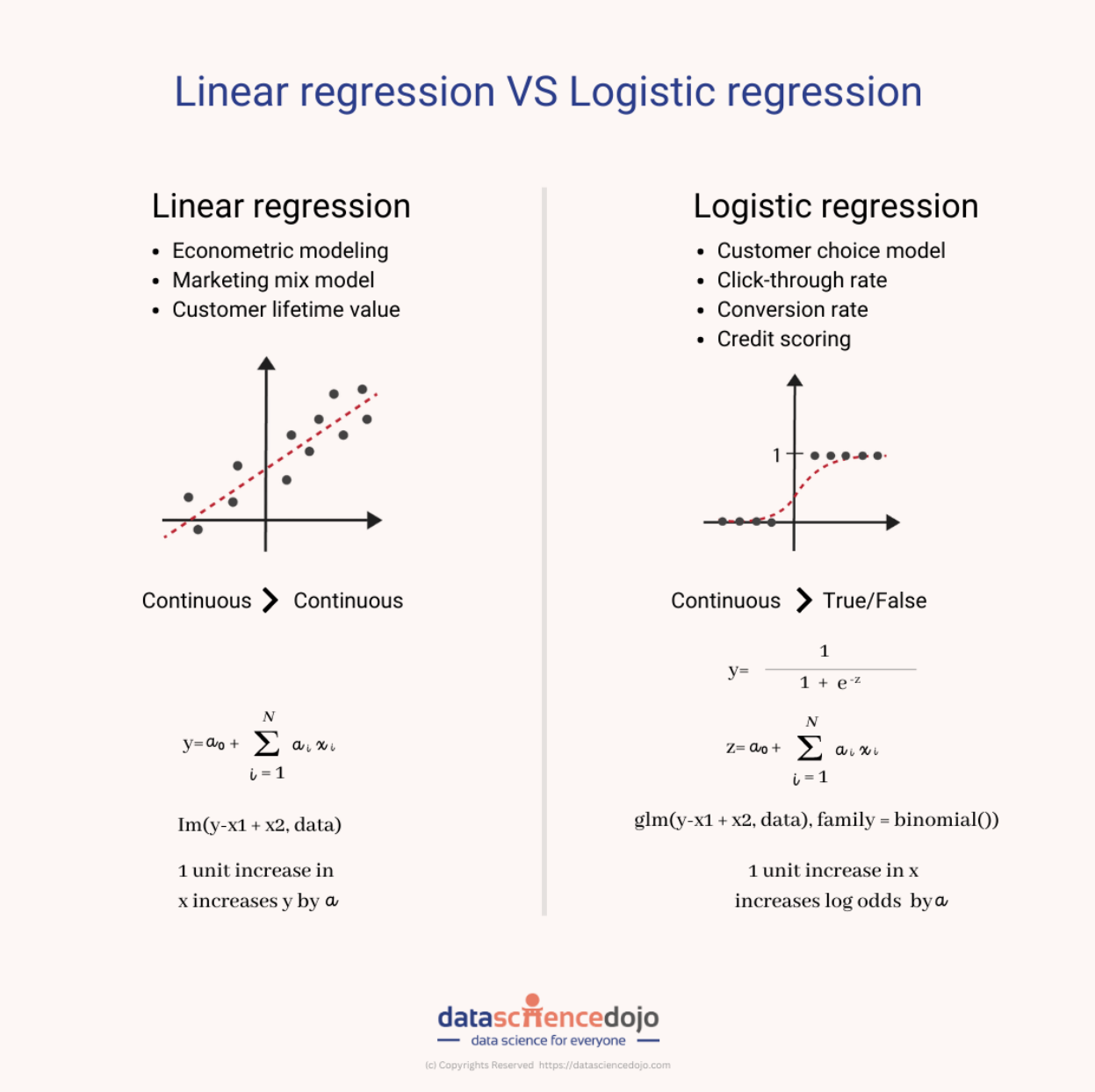
Indicators of Overfitting
Low Training Error, High Test Error: The model performs exceptionally well on the training data but poorly on the test data.

High Variance: The model's predictions vary significantly with different subsets of the training data.

How to Avoid Overfitting
Cross-Validation: Use techniques like k-fold cross-validation to ensure the model performs well on different subsets of the data.

Regularization: Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to penalize large coefficients and reduce model complexity.

Pruning: In decision trees, prune the tree to remove branches that have little importance.



Classification Problems

Classification problems are a category of machine learning problems where the goal is to categorize objects into predefined classes. In such problems, the algorithm is trained on a dataset that contains examples of the objects and their associated classes. After training, the algorithm can classify new, unseen objects into one of the classes.

There are various types of classification problems, including:

1. **Binary Classification:** The simplest form, where there are only two classes. Examples include email spam detection (spam or not spam) and medical diagnosis (sick or healthy).
2. **Multiclass Classification:** Involves more than two classes. An example is identifying the type of animal in an image (dog, cat, bird, etc.).
3. **Multilabel Classification:** Each object can belong to multiple classes. For example, in a movie categorization system, a movie might be labeled as both "action" and "adventure."
4. **Imbalanced Classification:** Occurs when the classes are not equally represented, like in fraud detection where fraudulent transactions are much less common than legitimate ones.
5. **Hierarchical Classification:** There's a hierarchy in the classes, like in a taxonomy of biological organisms.

Logistic Regression

Definition

A statistical method used for binary classification problems. It predicts the probability that a given instance belongs to a particular class.

Basic Principles

- **Binary Classification:** Used when the outcome variable is binary
- **Sigmoid Function:** The core of logistic regression is the sigmoid function, which maps any real-valued number into a value between 0 and 1, making it suitable for estimating probabilities.

Mathematical Model

- **Equation:** $P(Y = 1) = \frac{1}{1+e^{-(\beta_0+\beta_1 X)}}$ Can be expanded to include multiple features like B1X, B2X2, B3X3, ... BnXn --> (b0+b1x+b2x2+b3x3+bnxn)

  Where $P(Y = 1)$ is the probability of the instance being in class 1, X is the feature, β_0 and β_1 are the model coefficients.

Key Concepts

- **Odds and Log Odds**

    - **Odds:** Ratio of the probability of an event happening to the probability of it not happening.
    - **Log Odds (Logit):** The logarithm of the odds; logistic regression models this relationship.
- **Decision Boundary:**
    - A threshold is set (commonly 0.5) to decide the class of the instance. If the predicted probability is higher than the threshold, the instance is classified into one class, otherwise into the other. More about decision boundaries later.
- **Cost Function and Gradient Descent**
    - Unlike linear regression, logistic regression uses a different cost function (like cross-entropy) to accommodate the sigmoid function.
    - Gradient descent is used to minimize this cost function

Model Evaluation

- **Confusion Matrix:** A table used to describe the performance of a classification model

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- **Evaluation Metrics:** Accuracy, Precision, Recall, F-Score

Accuracy = (TP + TN) / (TP + TN + FP + FN), Precision = TP / (TP + FP), Recall = TP / (TP + FN)

- **ROC Curve and AUC:** Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) are used for evaluating the performance of binary classification models.

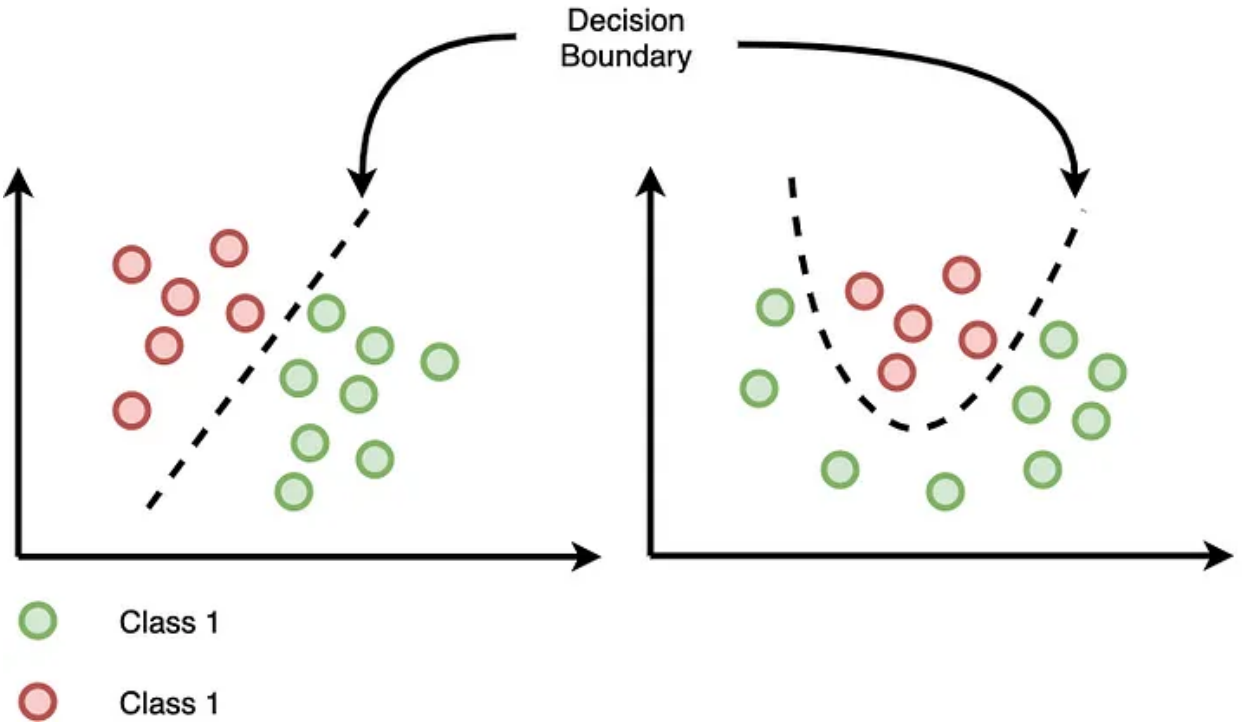
F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

ROC Curve: plots the true positive rate (recall) against the false positive rate.
AUC: single scalar value that summarizes the performance of the model

Challenges

- **Handling Imbalanced Data:** Requires specific strategies as logistic regression can be biased toward the majority class
- **Feature Scaling:** Similar to linear regression logistic regression benefits from feature scaling.
- **Interpreting Coefficients:** Understanding the impact of each feature on the goods can be less straightforward than in linear regression

Decision Boundaries

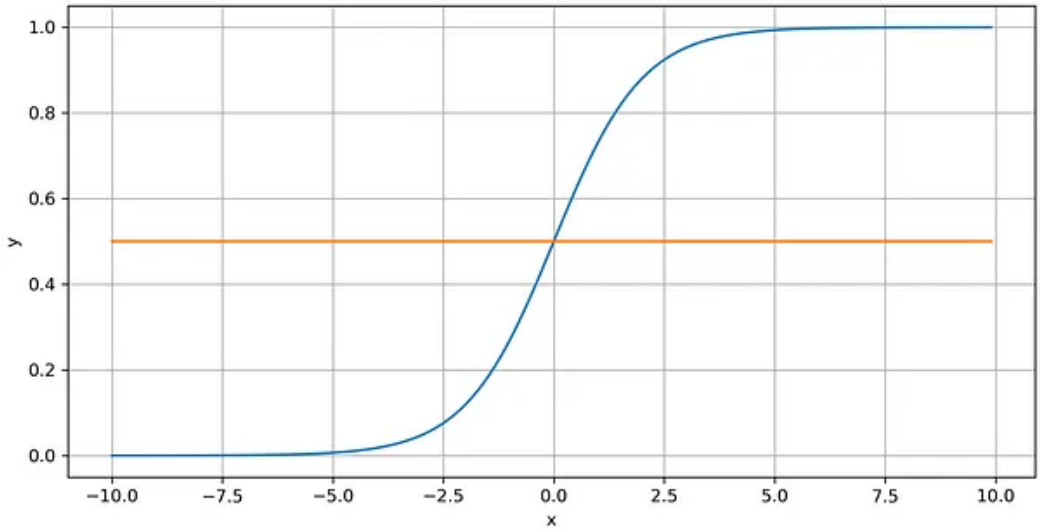


The fundamental application of logistic regression is to determine a decision boundary for a binary classification problem. Although the baseline is to identify a binary decision boundary, the approach can be very well applied for scenarios with multiple classification classes or multi-class classification.

In the above diagram, the dashed line can be identified as the decision boundary since we will observe instances of a different class on each side of the boundary. Our intention in logistic regression would be to decide on a proper fit to the decision boundary so that we will be able to predict which class a new feature set might correspond to. The interesting fact about logistic regression is the utilization of the sigmoid function as the target class estimator. Let us have a look at the intuition behind this decision.

The sigmoid function for parameter **z** can be represented as follows. Note that the function always lies in the range of 0 to 1, boundaries being asymptotic. This gives us a perfect output representation of probabilities too.

$$g(z) = \frac{1}{1 + e^{-z}}$$



Cost Function and Gradient Descent for Logistic Regression

In logistic regression, the goal is to find the optimal parameters that minimize the difference between predicted probabilities and actual class labels. This is where the cost function comes into play. The cost function helps us measure how well our model is performing and guides us toward finding the optimal parameters. You need to define a cost function.

The logistic regression cost function, also known as the log loss or cross-entropy loss, is a measure of the error between the predicted probabilities and true class labels. The logistic regression cost function is also known as the cross-entropy loss function or the log loss function. It is a convex function, which means that it

has a single global minimum. This makes it easier to optimize using gradient descent.

The logistic regression cost function possesses several desirable properties that make it suitable for training binary classifiers. These properties include:

1. Convexity

The cost function is convex, meaning it has a single global minimum. This allows us to use optimization algorithms like gradient descent to find the optimal parameters.

2. Continuous and Differentiable

The cost function is continuous and differentiable, which enables us to use gradient-based optimization techniques to minimize it efficiently.

3. Monotonicity

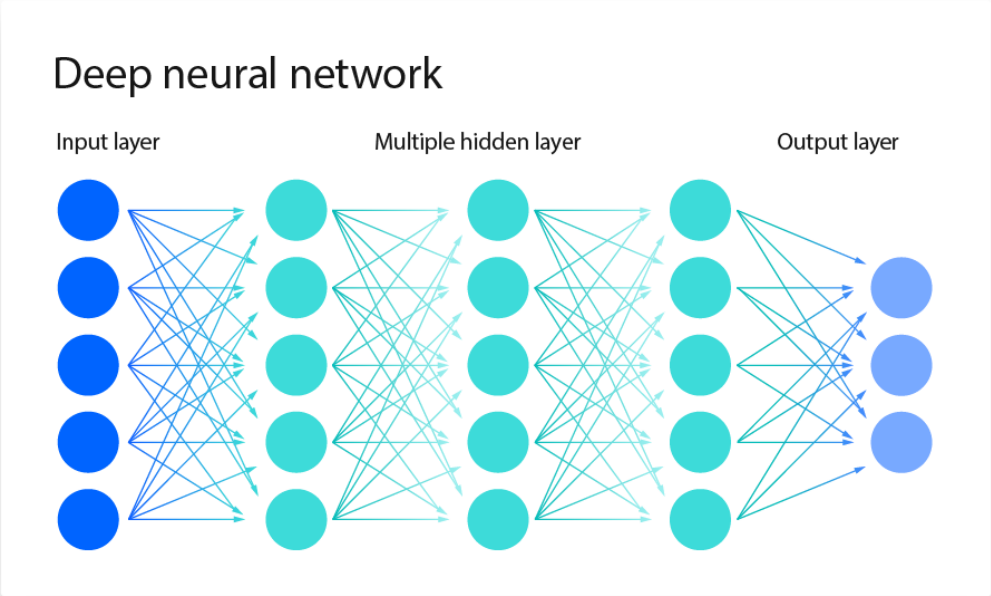
As we update the model parameters to minimize the cost function, it leads to an increase in model performance and a decrease in prediction errors.

Gradient descent is used with the cost function to find the optimal variables w and b so that it minimizes the cost function for the data-set. It is used to find the global or local minima of a differentiable function. To find the gradients you have to train multiple times and update the weights and bias to finally have the optimized values for weight and bias.

Overfitting and regularization

Overfitting on a very complex model can be avoided with regularization, which help reduce the number of features. Regularization help find our which features to remove from the model, as it applies a “penalty” to the input parameters with the larger coefficients, which limits the amount of variance in the model. There are multiple regularization methods such L1 regularization, Lasso regularization and dropout, which all seek to identify and reduce noise within the data.

Session 5 - Basics of Neural Networks



A Neural Network is interconnected layers of small units called nodes that perform mathematical operations to detect patterns in data. NN algorithms are built in a way that mimics how human neurons work. Neural networks consists of multiple things here are some of the important terms and definitions.

1. **Neuron** — This is a basic building block of a NN. It takes weighted values, performs mathematical calculation and produce output. It is also called a unit, node or perceptron.
2. **Input** — This is the data/values passed to the neurons.
    a. **Input Layer:** Receives the input signal to be processed
    b. **Hidden Layer:** Layers of nodes (neurons) between the input and output layer. Each layer applies weights to the inputs and passes them through an activation function
    c. **Output Layer:** Produces the final output of the network
3. **Deep Neural Network (DNN)** — This is an ANN with many hidden layers (layers between the input (first) layer and the output (last) layer).
4. **Weights** — These values explain the strength (degree of importance) of the connection between any two neurons.

5. **Bias** — is a constant value added to the sum of the product between input values and respective weights. It is used to accelerate or delay the activation of a given node.
6. **Activation function** — is a function used to introduce the non-linearity phenomenon into the NN system. This property will allow the network to learn more complex patterns.

Feed-Forward Neural Networks (FFNNs)

Definition

- A type of artificial neural network where connections between the nodes do not form cycles. The information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any), and to the output nodes.

Architecture

- **Layered Structure:** Consists of an input layer, one or more hidden layers and an output layer
- **No Backward or Lateral Connections:** Unlike recurrent neural networks, FFNNs have unidirectional connections.

Key Characteristics

- **Data Flow**
  - **Sequential Processing:** Information moves through the network layer by layer
  - **No Feedback Loops:** Different from recurrent neural networks (RNNs), where feedback loops allow for information to cycle through the network.
- **Hidden Layers:**
  - **Role:** These layers enable the network to learn complex patterns and perform sophisticated computations.
  - **Depth of the Network:** The number of hidden layers defines the “depth” of the network
- **Activation Functions in Hidden Layers**

- **Non-Linearity:** Functions like ReLU, Sigmoid or Tanh are used to introduce non-linearity, allowing the network to learn complex data patterns.
- **Influence on Learning:** The choice of activation function affects the network's ability to converge and the speed of learning.

- **Output Layer**
  - **Task-Specific Function:** The activation function in the output layer is chosen based on the specific task (e.g., softmax for multi-class classification, sigmoid for binary classification).

Training FFNNs

- **Backpropagation**
  - **Video**: https://www.youtube.com/watch?v=Ilg3gGewQ5U
  - **Mechanism:** A method used to calculate the gradient of the loss function with respect to each weight by the chain rule, moving backward through the network.
  - **Purpose:** To update the weights and biases in a manner that minimizes the loss function
- **Loss Functions**
  - **Selection:** Based on the task (e.g., cross-entropy loss for classification, mean squared error for regression).
- **Optimization Algorithms**
  - **Examples:** Gradient descent, stochastic gradient descent (SGD), Adam, etc.
  - **Function:** Used to minimize the loss function and update the weights

Evaluation and Performance

- **Evaluation Metrics**
  - **Dependent on Task:** Accuracy, precision, recall, F1-score for classification tasks; mean squared error, mean absolute error for regression tasks.
- **Overfitting and Generalization**

    - **Regularization Techniques:** Methods like dropout, L1/L2 regularization to prevent overfitting
    - **Cross-Validation**: Used to ensure the model generalizes well to new data
- **Model Tuning:**
    - **Hyperparameters:** Adjustments of learning rate, number of layers, number of neurons per layer, type of activation function, etc., to improve performance

Evaluation Metrics for Classification

Overview

Evaluation metrics are important for assessing the performance of classification models, helping to understand their accuracy, reliability and applicability to real-world scenarios.

Common Metrics

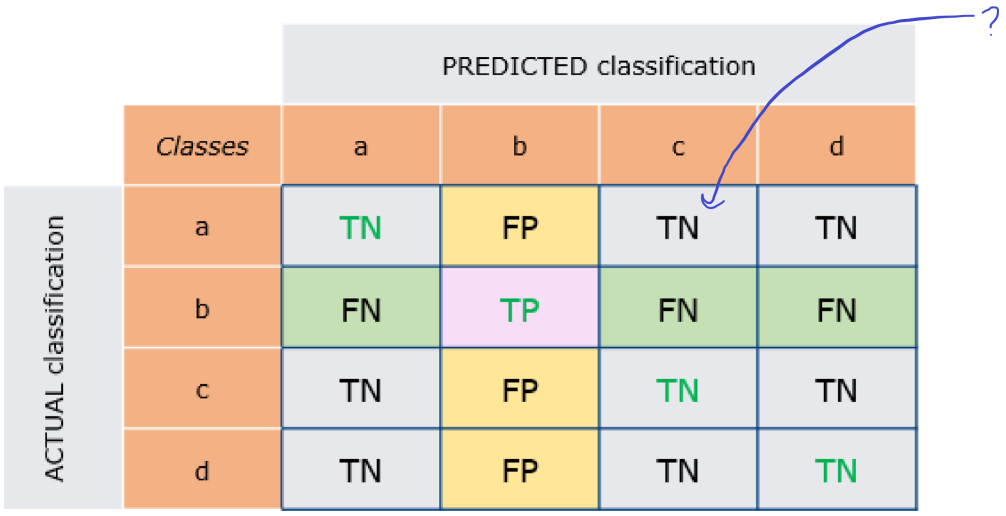
	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- **Accuracy**
    - **Definition**: The proportion of correctly predicted observations to the total observation
    - **Formula:** $Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$
- **Precision**
    - **Definition:** The proportion of positive identifications that were actually correct
    - **Formula:** $Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$
    - **Relevance:** Particularly important in scenarios where the cost of false positives is high.
- **Recall (Sensitivity)**
    - **Definition:** The proportion of actual positives that were correctly identified.
    - **Formula:** $Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$

- **Relevance:** Critical in situations where missing a positive is costly (e.g., disease diagnosis)

- **F1-Score**
  - **Definition**: The harmonic mean of precision and recall.
  - **Formula:** $F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$
  - **Relevance:** Useful when you need a balance between precision and recall

- **Confusion Matrix**
  - A table layout that allows visualization of the performance of an algorithm.
  - **Components:** True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN)

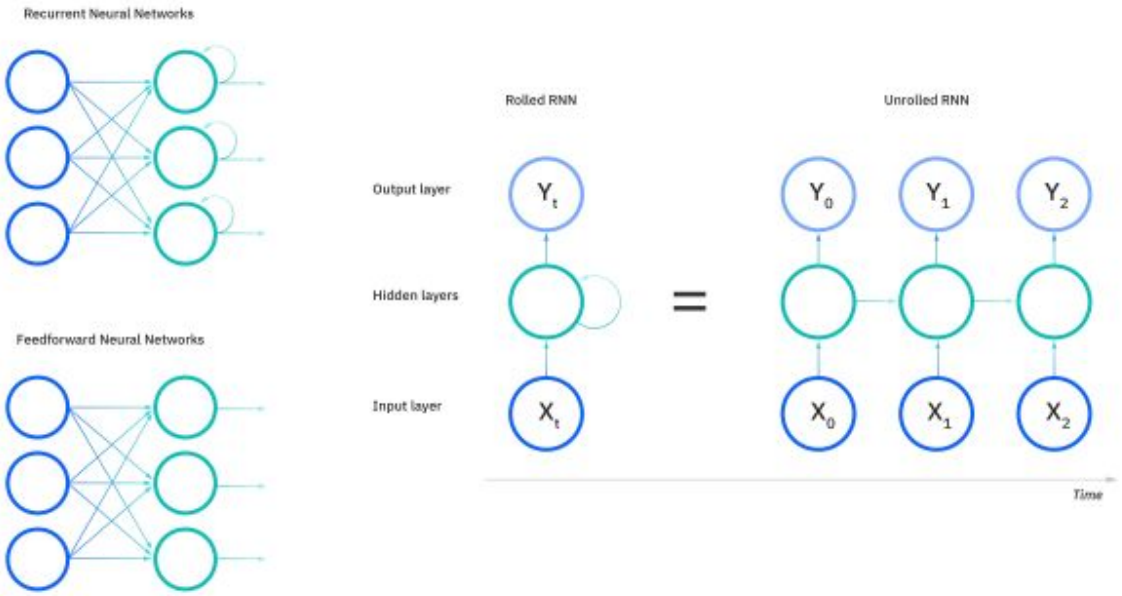


Considerations

- **Choice of Metric:** Depends on the specific requirements and trade-offs of the classification task.
- **Balanced vs. Imbalanced Datasets:** In imbalanced datasets, accuracy alone can be misleading.

- **Threshold Selection:** The choice of threshold for classifying probabilities into class labels can impact the performance and should be tuned based on the problem context.
- **Use Multiple Metrics:** Relying on a single can give a skewed view of the model's performance. It's often beneficial to evaluate the model using multiple metrics

Session 6 - Recurrent Neural Networks



A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition and image captioning. Like feedforward and convolutional neural networks (CNN), RNN utilize training data to learn. RNN takes information from prior inputs to influence current input and output.

While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also be helpful in

determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions.

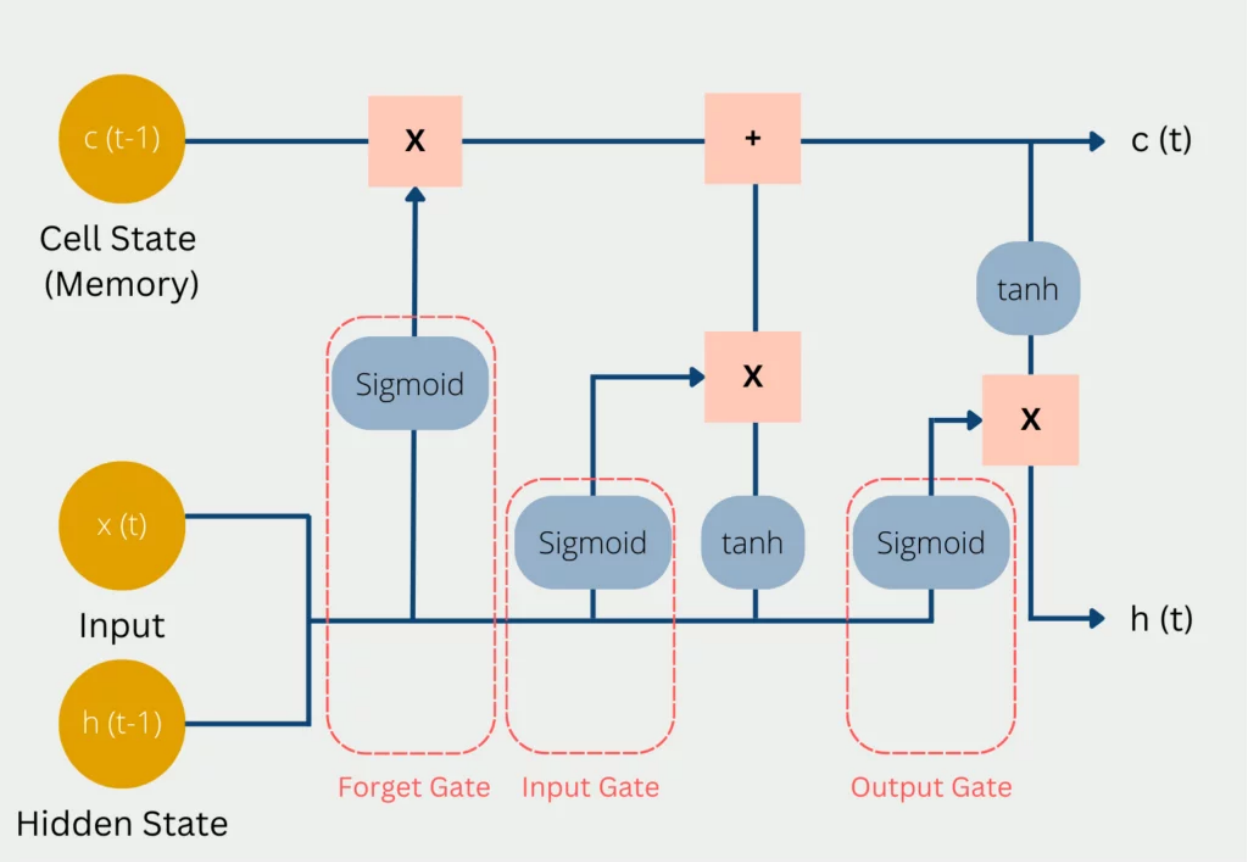
Let's take an idiom, such as "feeling under the weather", which is commonly used when someone is ill, to aid us in the explanation of RNNs. In order for the idiom to make sense, it needs to be expressed in that specific order. As a result, recurrent networks need to account for the position of each word in the idiom and they use that information to predict the next word in the sequence.

Another distinguishing characteristic of recurrent networks is that they share parameters across each layer of the network. While feedforward networks have different weights across each node, recurrent neural networks share the same weight parameter within each layer of the network. That said, these weights are still adjusted in the through the processes of backpropagation and gradient descent to facilitate reinforcement learning.

Recurrent neural networks leverage backpropagation through time (BPTT) algorithm to determine the gradients, which is slightly different from traditional backpropagation as it is specific to sequence data. The principles of BPTT are the same as traditional backpropagation, where the model trains itself by calculating errors from its output layer to its input layer. These calculations allow us to adjust and fit the parameters of the model appropriately. BPTT differs from the traditional approach in that BPTT sums errors at each time step whereas feedforward networks do not need to sum errors as they do not share parameters across each layer.

Through this process, RNNs tend to run into two problems, known as exploding gradients and vanishing gradients. These issues are defined by the size of the gradient, which is the slope of the loss function along the error curve. When the gradient is too small, it continues to become smaller, updating the weight parameters until they become insignificant—i.e. 0. When that occurs, the algorithm is no longer learning. Exploding gradients occur when the gradient is too large, creating an unstable model. In this case, the model weights will grow too large, and they will eventually be represented as NaN. One solution to these issues is to reduce the number of hidden layers within the neural network, eliminating some of the complexity in the RNN model.

Long Short-Term Memory (LSTMs)



LSTMs are a special kind of Recurrent Neural Network (RNN) capable of learning long-term dependencies. They were developed to overcome the problem of short-term memory in traditional RNNs.

Key Features

- **Memory Cell**
  - **Core Component:** Acts as a ‘memory storage’ within the network.
  - **Function:** Stores and outputs information, with the ability to add or remove information via gates
- **Gates in LTSM**
  - **Forget Gate:** Decides what information should be thrown away from the cell state.
  - **Input Gate:** Updates the cell state with new information.
  - **Output Gate:** Determines what the next hidden state should be.

Structure and Mechanism

- **Cell State:** The ‘conveyor belt’ of the network carries relevant information throughout the processing of the sequence.
- **Hidden State:** Represents the current output of the LTSM cell, influenced by the previous hidden state and the current input.
- **Gate Operations**
  - **Sigmoid Layer:** Controls which parts of the cell state are output.
  - **Tanh Layer:** Creates a vector of new candidate values that could be added to the state.

Learning Process

- **Backpropagation Through Time (BPTT)**
  - **Mechanism:** Used to train LSTM networks, addressing the vanishing gradient problem common in standard RNNs
  - **Adjustments:** Involves the careful tuning of parameters to maintain the balance between long-term and short-term memory
- **Gradient Flow**
  - **Strength:** LSTMs allow gradients to flow unchanged, preventing the vanishing gradient problem.

Application

- **Sequence Prediction:** Ideal for tune series prediction, language modeling, text generation, etc.
- **Natural Language Processing:** Used in tasks like machine translation, speech recognition and sentiment analysis.

Advantages

- **Long-Term Dependency Learning:** Efficiently captures dependencies over long sequences.
- **Flexibility:** Adapts to various types of sequence data.

Challenges

- **Complexity:** More complex than traditional RNNs, leading to higher computational cost.
- **Parameter Tuning:** Requires careful tuning of parameters like learning rate, number of layers, etc.

Session 7 - Representation Learning

Input Representations

Overview

Input presentations are methods of converting raw data into a format that machine learning models can understand and process efficiently.

Key Aspects

- **Data Types**
  - **Variety:** Includes text, images, audio, video and more.
  - **Transformation:** Each type requires specific methods to transform into a suitable representation for machine learning models.
- **Text Data**
  - **Tokenization:** Breaking down text into words, characters or other meaningful elements (tokens)
  - **Vectorization Methods**
    - **One-Hot Encoding:** Represents each token as a unique vector
    - **TF-IDF (Team Frequency-Inverse Document Frequency):** Reflects the importance of a token in a collection of documents
    - **Word Embedding (e.g. Word2Vec, GloVe):** Maps words into continuous vector spaces where semantically similar words are mapped to nearby points.
- **Image Data:**
  - **Pixel Values:** Representing images as arrays of pixel values

    - **Feature Extraction:** Using techniques like edge section, color histogram, etc.
    - **Deep Learning Approaches:** Utilizing convolutional neural networks (CNNs) to learn hierarchical representations
- **Audio Data:**
    - **Spectrogram:** Visual representations of the spectrum of frequencies in a sound.
    - **MFCCs (Mel Frequency Cepstral Coefficients):** Capture the timbral aspects of audio
- **Video Data:**
    - **Frame Extraction:** Treating each frame as an individual image.
    - **Motion Features:** Capturing movement and changes over time.

Contextual Information-Based Representations

Focus on capturing the context in which data appears, which is especially important in areas like language processing, where the meaning of a word can depend heavily on its surrounding words.

Key Aspects:

- **Language Processing**
    - **Word Context:** Understanding the meaning of a word based on the words around it.
    - **Sentence Structure:** Capturing the grammatical and logical structure of sentences
- **Representation Techniques**
    - **Word Embeddings**
        - **Contextual Embeddings (e.g., BERT, ELMo):** Word representations that consider the context in which a word appears, offering dynamic embeddings as opposed to static ones.
    - **Sequence Models**

        - **RNNs, LSTMs, GRUs:** Used for capturing sequential information in text data.
        - **Attention Mechanisms:** Allow models to focus on relevant parts of the input sequence, improving the capture of context
- **Feature Extraction from Context**
    - **Co-occurrence Matrices:** Representing words based on their co-occurrence frequencies in the data.
    - **N-Grams:** Capturing local context by considering contiguous sequences of N items from a given sample of text or speech.

Applications

- **Natural Language Processing (NLP):** Essential for complex NLP tasks such of context-sensitive language modeling, text classification and
- **Speech Recognition:** Used to understand the context in spoken language for more accurate recognition.

Challenges

- **Complexity:** These methods can be computationally intensive and complex to implement
- **Data Requirements:** Often require large amounts of data to effectively capture contextual nuances.

Distributional Hypothesis

Definition

The Distributional Hypothesis posits that words occurring in similar contexts tend to have similar meanings. Based on the idea that the usage and meaning of a word can be understood from the company it keeps.

Key Aspects

- **Contextual Similarity**
    - Words are semantically similar to the extent that they share similar linguistic contexts.

    - Meaning is inferred from patterns of co-occurrence with other words.
- **Word Embeddings**
    - **Application:** Forms the basis for creating word embedding in NLP
    - **Techniques like Word2Vec and GloVe**
        - Utilize the distributional properties of words to embed them in continuous vector spaces.
        - Words with similar contexts are placed closer in vector space
- **Semantic Relationships**
    - Capturing not just similarity but also a range of semantic relationships (e.g., synonyms, antonyms) based on word co-occurrence patterns.

Importance in Machine Learning

- **Improved Language Models:** Leads to more effective representation of words for various NLP tasks.
- **Enhanced Understanding:** Facilitates a deeper understanding of language, capturing nuances and variations in meaning

Consideration

- **Corpus Quality:** The quality and diversity of the text corpus greatly influence the effectiveness of representations based on the Distributional Hypothesis.
- **Dimensionality Reduction:** Techniques like PCA may be used to reduce the dimensionality of word vectors

Self-Supervised Learning

Definition

A type of machine learning where the system learns to predict part of its input from other parts of its input using a learning signal derived from the data itself, rather than from external labels.

Key Features

- **Learning from the Data Itself**

    - Utilizes the structure within the data to generate labels and learn from them
    - Reduces reliance on human-labeled datasets.
- **Automatic Label Generation**
    - The algorithm creates its own labels from the input data, often by hiding some part of the data and learning to predict it

Methodology

- **Pretext Tasks**
    - Designed to create a learning scenario for the model to solve and learn useful representations
    - **Example:** Predicting the next word in a sentence, colonizing black and white images, solving jigsaw puzzles of images
- **Data Augmentation**
    - Manipulating input data to create ‘new’ data, which can be used for training
    - Enhances the model’s ability to generalize

Advantages

- **Efficiency:** More efficient use of data, leveraging unlabeled data which is often more abundant
- **Generalization:** Tends to learn more generalizable features.

Session 10 - Uncertainty Estimation, Out-of-Distribution Data and Robustness in Machine Learning

Inference Under Uncertainty

Deals with making predictions or decisions based on incomplete or uncertain information

Types of Uncertainty

- **Aleatoric Uncertainty**

    - Inherent variability in the data
    - **Example:** Sensor noise or inherent randomness in the system
- **Epistemic Uncertainty**
    - Due to lack of knowledge or information
    - Can be reduced with more data or better models
- **Key Approaches**
    - **Probabilistic Models**
        - Use probability distributions to model and reason about uncertainties
        - **Example:** Bayesian Neural Networks, Gaussian Processes
    - **Ensemble Methods**
        - Combine predictions from multiple models to estimate uncertainty
        - **Example:** Random Forests, ensemble of deep learning models
        - **Video:** https://www.youtube.com/watch?v=Un9zObFjBH0

Importance in Machine Learning

- Enhances the reliability and robustness of predictions
- Critical for design-making processes in uncertain environment

Tools and Techniques

- **Bayesian Methods:** For incorporating uncertainty into model parameters
- **Dropout as a Bayesian Approximation:** Using dropout layers in neural networks to estimate uncertainty

Model-Based Decision Making

Definition

Involves using machine learning models to inform and optimize decision-making processes

Key Principles

- **Integration of Predictive Models**

    - Utilizing models to predict outcomes or consequences of different decisions.
- **Risk Assessment**
    - Evaluating potential risks associated with decisions based on model predictions.
- **Optimization**
    - Applying algorithms to find the best decision, often considering multiple objectives and constraints.

Steps in Model-Based Decision Making

1. **Model Development**
    a. Building accurate and reliable predictive models based on historical data
2. **Scenario Analysis**
    a. Simulating different scenarios to understand possible outcomes of decisions
3. **Decision Analysis**
    a. Evaluating the effectiveness and impact of different decision options
4. **Continuous Learning**
    a. Updating models with new data to refine decision-making over time

Techniques

- **Monte Carlo Simulations:** For risk assessment and scenario analysis
- **Multi-Criteria Decision Analysis (MCDA):** Evaluating decisions based on multiple criteria

Out-of-Distribution Data

Definition

Refers to data that significantly differs from the training dataset on which a model was developed

Characteristics

- **Deviation from Training Set**
  - Presents patterns or characteristics not seen during training data
- **Unseen Variations**
  - Can include novel features or combinations of features

Impact on Machine Learning Models

- **Performance Degradation**
  - Models may perform poorly due to a lack of prior exposure to such data
- **Reliability Issues**
  - Can lead to expected or incorrect predictions

Strategies for Improvement

- **Data Augmentation**
  - Expanding the training set with synthetic and modified data to cover a broader range of scenarios
- **Domain Adaption**
  - Adjusting the model to perform well on a different but related data distribution
- **Regularization Techniques**
  - Enhancing the model's ability to generalize to unseen data

Robustness in Machine Learning

Refers to the ability of a machine learning model to maintain performance across a range of inputs and conditions, including noisy data and adversarial attacks.

Aspects of Robustness

- **Generalization**
  - The model's ability to perform well on unseen data similar to the training set
- **Noise Tolerance**

- Maintaining performance in the presence of random noise in the data

Enhancing Robustness

- **Data Augmentation**
  - Introducing variations in training data to mimic different scenarios
- **Regularization Techniques**
  - Methods like dropout, L1/L2 regularization to prevent overfitting.
- **Model Ensemble**
  - Combining multiple models to improve overall robustness

Evaluation

- **Stress Testing**
  - Evaluating model performance under extreme or challenging conditions

Challenges

- **Trade-off with Accuracy**
  - Improving robustness might sometimes reduce model accuracy on clean data.
- **Computational Complexity**
  - Some robustness-enhancing techniques increase computational demands

Ensembles of Neural Networks

Definition

Combining multiple neural network models to improve overall performance and robustness

Techniques

- **Bagging**
  - Training multiple of the same model independently on different subsets of the data
- **Boosting**

    - Sequentially training models, where each model learns from the errors of the previous
- **Stacking**
    - Combining predictions from multiple models using to make final predictions (e.g. CNN + LR)

Advantages

- **Improved Accuracy:** Often achieves better performance than individual models
- **Reduced Overfitting:** Ensemble methods can reduce the risk of overfitting by averaging out biases
- **Robustness:** Increase the robustness of the system against noise and outliers

Challenges

- **Computational Complexity:** Higher resource requirements due to training multiple models
- **Model Coordination:** Managing and combining multiple models can be complex
- **Parameter Tuning:** Requires careful tuning of parameters for each model in the ensemble

Evaluation

- **Cross-Validation:** Used to assess the performance of ensemble models
- **Performance Metrics:** Accuracy, precision, recall, F1-score, etc., depending on the task

Bayesian Methods

Bayesian methods in machine learning involve using Bayes' Theorem to update the probability estimate for a hypothesis as more evidence or information becomes available

Key Concepts

- **Bayes' Theorem**

    - Fundamental to Bayesian methods, providing a way to update the probability of a hypothesis based on new data
- **Prior Probability**
- **Posterior Probability**
- **Likelihood**

Advantages

- **Incorporation of Prior Knowledge:** Allows the integration of expert knowledge or previous information
- **Probabilistic Interpretation:** Offers a probabilistic framework for model predictions and uncertainty estimation

Techniques

- **Bayesian Neural Networks**
    - Neural networks where the weights are treated as random variables with probability distributions
- **Markov Chain Monte Carlo (MCMC)**
    - A method for sampling from the posterior distribution when it cannot be computed directly
- **Variational Inference**
    - An approach to approximating complex probability distributions through optimization

Challenges

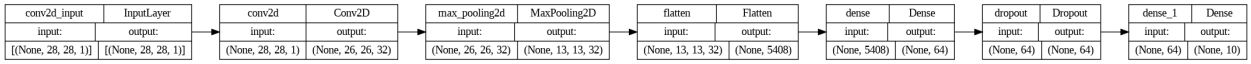
- **Computational Complexity:** Bayesian methods can be computationally intensive, especially for large datasets
- **Choice of Priors:** The selection of prior distributions can significantly impact model outcomes

Model Evaluation

- **Predictive Performance:** Assessed using metrics like accuracy, precision, recall depending on the task

- **Uncertainty Quantification**
  - Evaluating the quality and reliability of the uncertainty estimates provided by the model

Capstone Project



Convolutional Neural Network (CNN) Model Structure

- **Convolutional Layer:** The first layer is a 2D convolutional layer with 32 filters of size 3×3 and `relu` activation function, suitable for processing the 28×28 input images.
- **Pooling Layer:** A MaxPooling2D layer follows with a 2×2 window, reducing the spatial dimensions of the output from the previous layer.
- **Flattening:** The output is then flattened to feed into the dense layers.
- **Dense Layer:** A fully connected layer with 64 units and `relu` activation function is used for learning non-linear combinations.
- **Dropout:** To prevent overfitting, a dropout layer with a rate of `0.5` is included, which randomly sets a fraction of input units to 0 during training.
- **Output Layer:** The final layer is a dense layer with 10 units (for 10 classes) and a `softmax` activation function for classification.

Model Compilation

- **Optimizer:** The Adam optimizer with a learning rate of `0.0001` is chosen for efficient and adaptive learning rate adjustments.
- **Loss Function:** Sparse Categorical Cross entropy is used as it is suitable for multi-class classification tasks where each class is mutually exclusive.
- **Metrics:** The model's performance is evaluated based on accuracy

Model Training

- **Dataset:** The model is trained on pre-processed grayscale images.
- **Epochs:** The training runs for 10 epochs, where an epoch is one complete pass through the entire training dataset.
- **Batch Size:** A batch size of 32 is used, which means 32 samples are used to estimate the error gradient before the model weights are updated.
- **Validation:** The model is validated on a separate validation dataset to monitor and prevent overfitting.

Regularization Techniques

- **Dropout:** As mentioned, dropout is used to randomly ignore a subset of neurons during training, which helps in preventing overfitting.
- **Learning Rate:** A relatively small learning rate ensures that the model doesn't converge too quickly to a suboptimal solution.

What is ReLU (Rectified Linear Unit)?

It is a rectified linear activation function, which is a linear function that will output the input directly if it is positive otherwise it will output zero. It is a default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance. ReLU is a default activation for convolutional neural networks. It overcomes the vanishing gradient problem.

What is Softmax?

The softmax converts a vector into a probability distribution of multiple possible outcomes of that vector. Softmax is a generalization of the logistic function to multiple dimensions.

What is Adam optimizer?

Adam optimizer is implements the Adam algorithm, which is a variation of a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments.