Slide Report Week 3

Le Viet Hung - AI Intern

Content

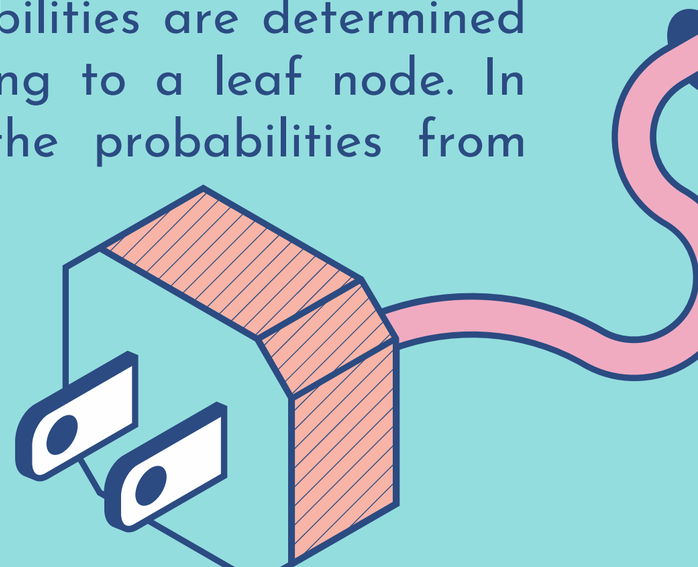
1. Estimating Class Probabilities
2. Gini Impurity or Entropy
3. The CART Training Algorithm
4. Computational Complexity
5. Regularization Hyperparameters

1.Estimating Class Probabilities

Estimating class probabilities refers to the process of predicting the probabilities of different classes or categories that a data point belongs to.

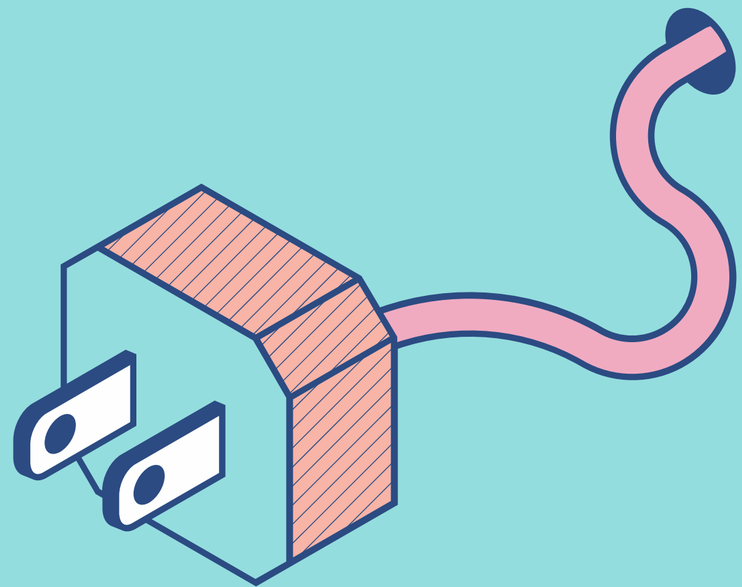
There are various techniques to estimate class probabilities depending on the algorithm used. Some common methods include:

- Logistic Regression: In logistic regression, the model outputs the probability of a data point belonging to the positive class using the sigmoid function. The predicted probabilities lie between 0 and 1.
- Decision Trees and Random Forests: For decision trees, the class probabilities are determined by the proportion of training samples of a particular class that belong to a leaf node. In random forests, the class probabilities are obtained by averaging the probabilities from individual decision trees.



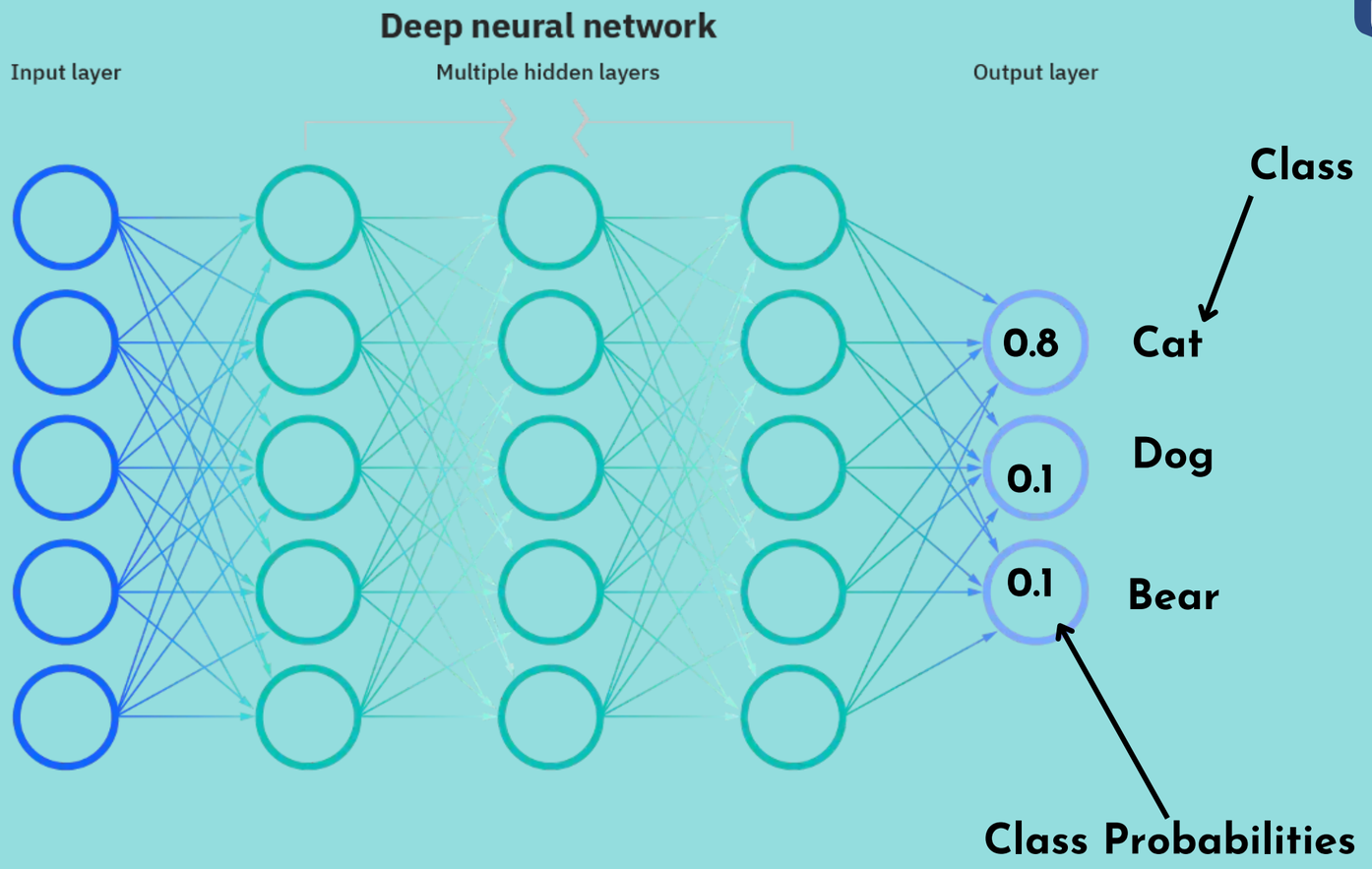
1.Estimating Class Probabilities

- Support Vector Machines (SVM): For binary classification using SVM, the model outputs the signed distance of a data point from the decision boundary, and this distance can be converted into class probabilities using calibration techniques like Platt scaling.
- Neural Networks: In deep learning models like neural networks, the output layer typically employs a softmax activation function, which converts the raw outputs into probabilities. Each neuron in the output layer represents the probability of the corresponding class.
- Naive Bayes: In Naive Bayes classifiers, class probabilities are estimated using Bayes' theorem, assuming that the features are conditionally independent given the class.



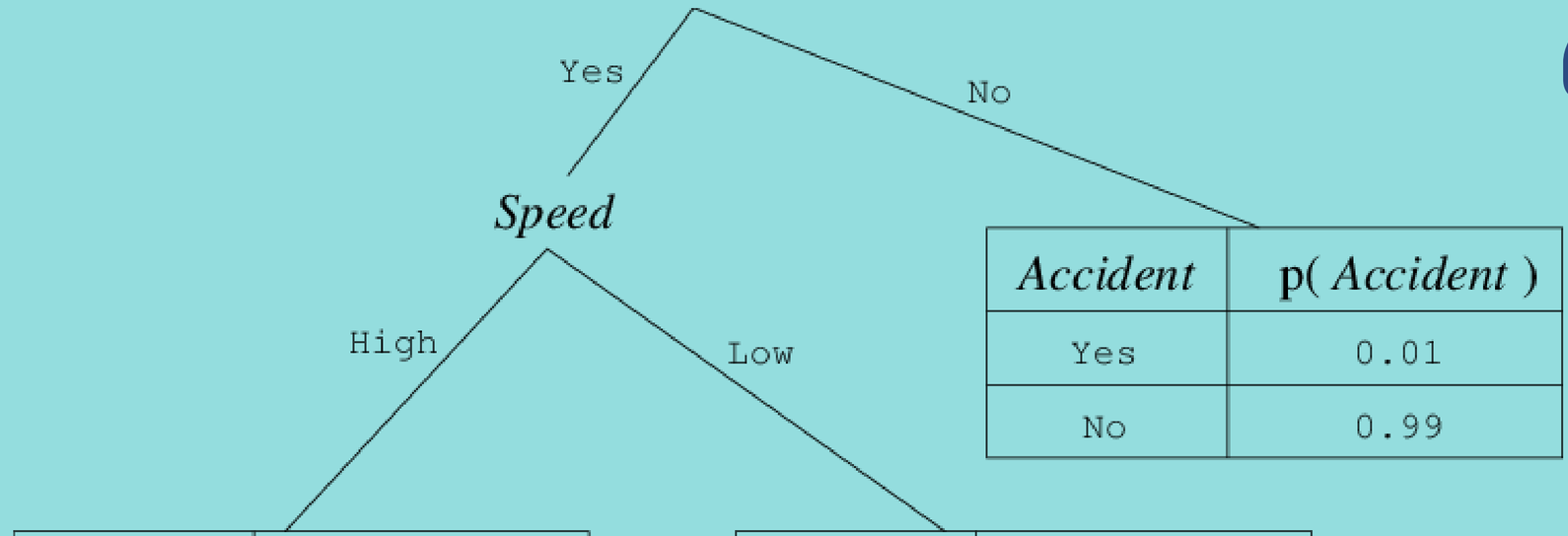
1.Estimating Class Probabilities

Example: Neural Networks



1.Estimating Class Probabilities

Example: Decision Tree



Accident	p(*Accident*)
Yes	0.01
No	0.99

Accident	p(*Accident*)
Yes	0.13
No	0.87

Accident	p(*Accident*)
Yes	0.02
No	0.98

Class probabilities

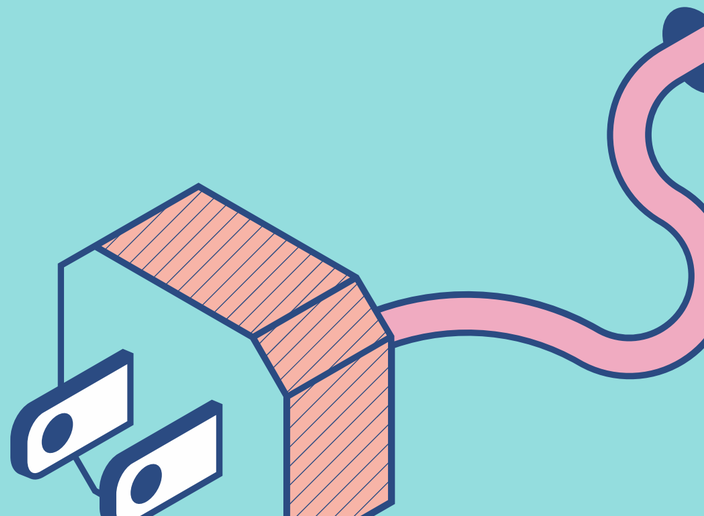
2.Gini Impurity or Entropy

Gini Impurity is a measure of impurity or uncertainty in a dataset. It measures the level of disorder or uncertainty in the dataset.

If a dataset contains samples from multiple different classes, the entropy will be high.

Consider a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as p_i. Then the Gini Impurity of D is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

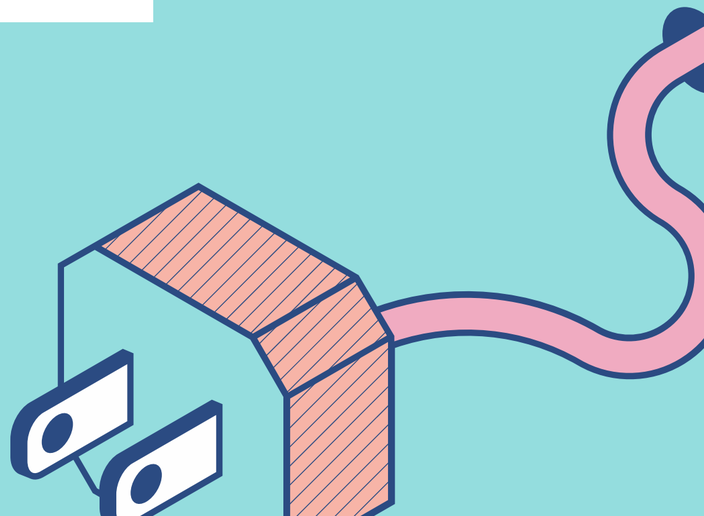


2.Gini Impurity or Entropy

Entropy is a measure of impurity or uncertainty in a dataset. It measures the level of disorder or uncertainty in the dataset.

The probability distribution of a random variable x can get values $x_1, x_2, \ldots, x_n$. So the probability x get the value x_i is $p_i = p(x = x_i)$. $0 \leq p_i \leq 1$, $\sum_{i=1}^{n} p_i = 1$. This probability is assigned as $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. So the Entropy of the distribution $\mathbf{p}$ is defined as:

$$H(\mathbf{p}) = -\sum_{i=1}^{n} p_i \log(p_i)$$



2.Gini Impurity or Entropy

Example: Here's a table demonstrating the computation of Gini Impurity and Entropy for a binary classification problem with two classes (Class A and Class B)
Assume we have a dataset with 20 samples, where 14 samples belong to Class A and 6 samples belong to Class B.

Class	Count (m)	Proportion (p)	Gini Impurity (Gini)	Entropy
A	14	14/20 = 0.7	0.7*(1-0.7) = 0.21	-0.7*log2(0.7) = 0.609
B	6	6/20 = 0.3	0.3*(1-0.3) = 0.21	-0.3*log2(0.3) = 0.442
Total	20	1	Gini = 0.21 + 0.21 = 0.42	Entropy = 0.609 + 0.442 = 1.051

- "Count (m)" represents the number of samples in each class.
- "Proportion (p)" represents the proportion of each class in the dataset (Count / Total).
- "Gini Impurity (Gini)" is calculated as p*(1-p) for each class.
- "Entropy" is calculated as -p*log(p) for each class.

2.Gini Impurity or Entropy

Comparision:

	Gini Impurity	Entropy
Computation	Less expensive	More expensive
Sensitivity to Class Imbalance	Less sensitive	More sensitive

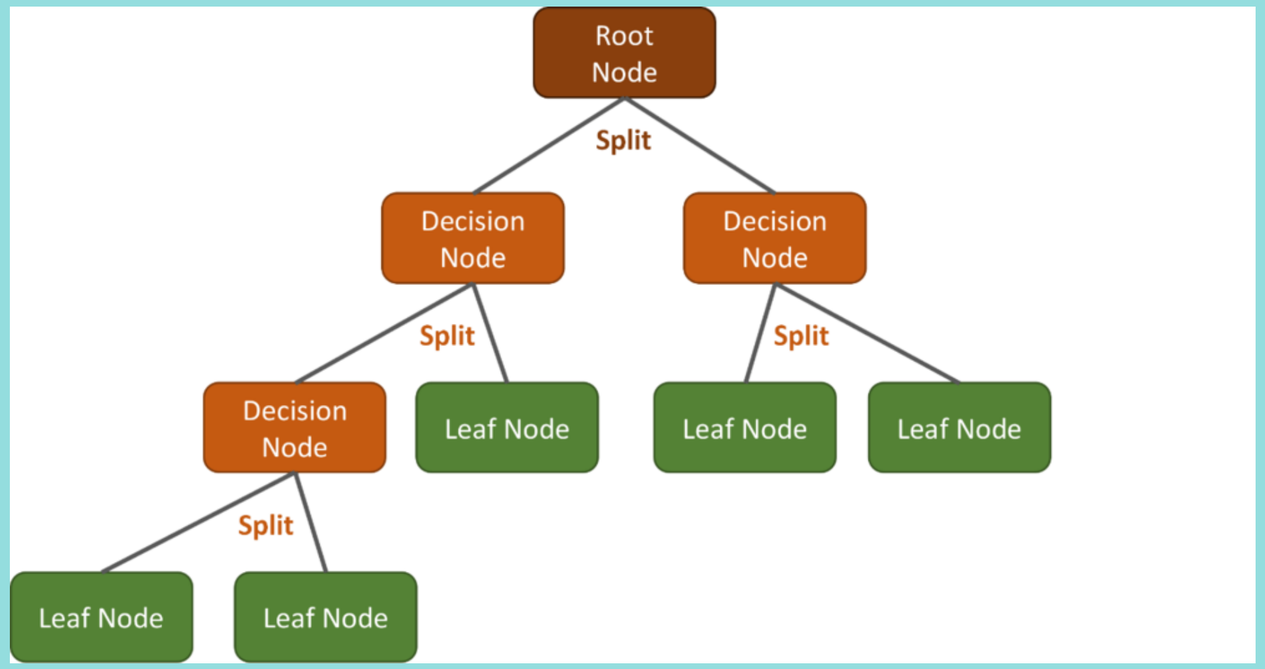
3.The CART Training Algorithm

Classification and Regression Trees (CART) is a variation of Decision Tree algorithms that can be used for classification or regression predictive modeling problems.

3.The CART Training Algorithm

One of the Difference to Decision Tree is Slitting Strategy:
CART Algorithm: Binary splits (Two child nodes per internal node)
Decision Tree: Binary or multi-way splits (Multiple child nodes per internal node)

Visual Representation of CART Algorithm:



4. Computational Complexity

Computational complexity is an important aspect of machine learning that refers to the efficiency of algorithms used to solve specific tasks. It deals with understanding the amount of computational resources, such as time and memory, required by an algorithm as a function of the input size.

In machine learning, the computational complexity include both of *time complexity* and *space complexity*.

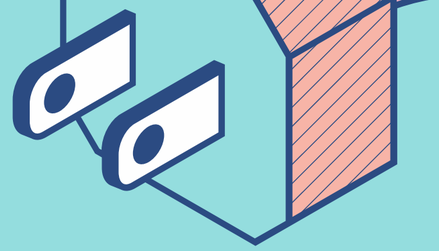
Space Complexity: refers to the amount of memory required by an algorithm to execute as a function of the input size

Time Complexity: refers to the amount of time required for an algorithm to complete its task as a function of the input size

4. Computational Complexity

- **Linear Regression:** The time and space complexity of linear regression is typically O(n), where "n" is the number of data points. This makes it very efficient for large datasets.
- **Support Vector Machines (SVM):** The time complexity of SVM depends on the chosen kernel and the optimization algorithm used. In general, it ranges from O(n^2) to O(n^3) for training and O(m) for prediction, where "n" is the number of data points and "m" is the number of support vectors.
- **Decision Trees:** The time complexity for training decision trees is generally O(n * d * log(n)), where "n" is the number of data points and "d" is the number of features. The space complexity is O(1) during training, but it can be O(n) for prediction if the tree is fully grown.
- **Neural Networks:** The time complexity of training neural networks depends on the architecture and the optimization algorithm used. Deep neural networks with many layers can be computationally intensive, often requiring significant computational resources and time.
- **k-Nearest Neighbors (k-NN):** The time complexity of k-NN for training is O(1) as it simply memorizes the data points. However, the time complexity for prediction is O(n), where "n" is the number of data points, as it needs to compute distances to all data points in the training set.

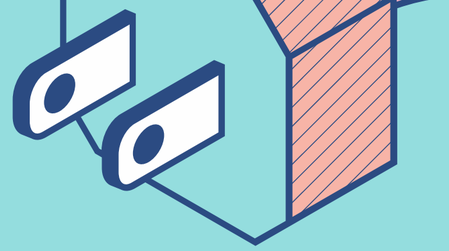
5. Regularization Hyperparameters



Regularization hyperparameters are a technique used in machine learning, including deep learning, to prevent overfitting and improve the generalization performance of a model.

Regularization is a form of adding a penalty term to the model's objective function during training. The penalty term discourages the model from learning overly complex patterns that might fit the training data well but fail to generalize to unseen data.

5. Regularization Hyperparameters

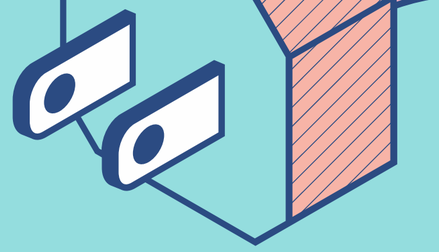


Techniques in Regularization Hyperparameters:

In Linear Regression:

- L1 Regularization (Lasso Regularization):
  - Example: In linear regression, the cost function with L1 regularization can be written as: Cost = Mean Squared Error + λ * ∑|weights| where λ is the regularization strength parameter. This will encourage some weights to become exactly zero, effectively performing feature selection.
- L2 Regularization (Ridge Regularization):
  - Example: In logistic regression, the cost function with L2 regularization can be written as: Cost = Cross-Entropy Loss + λ * ∑(weights^2) where λ is the regularization strength parameter. This will spread the weight values more evenly across features and prevent large weight values.

5. Regularization Hyperparameters



Techniques in Regularization Hyperparameters:

In Deep Learning:

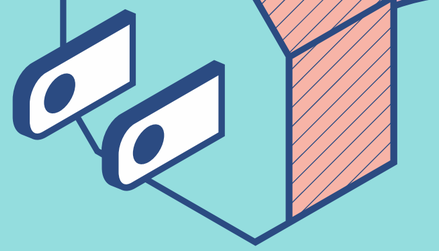
- Dropout:

  Example: In deep neural networks, dropout is applied during training, where some neurons are randomly deactivated with a certain probability. This helps prevent overfitting and encourages the network to learn more robust features.

- Data Augmentation:

  Example: In image classification tasks, data augmentation involves applying random transformations (e.g., rotations, translations, flips) to the training images. This increases the effective size of the training dataset and makes the model more invariant to variations in the input data.

5. Regularization Hyperparameters



Techniques in Regularization Hyperparameters:

In Deep Learning:

- Early Stopping:

  Example: During training, monitor the model's performance on a validation set. When the validation loss starts increasing or stagnating, stop the training process to prevent overfitting on the training data.

- Batch Normalization:

  Example: Batch normalization is a technique that normalizes the inputs to a layer in a deep neural network during training. This helps stabilize and speed up training and acts as a form of regularization.

Thank you