Statistical To Neural Network - A Tale of Evolution

Aditya Jha[1], Tirthraj Dutta Roy [2], and Altamash Amil[3]

[1]aditya01jha.02@gmail.com
[2]tirthrajdroy@gmail.com
[3]altamashamilap@gmail.com

November 11, 2024

Abstract

This paper provides a brief overview of machine learning, three three primary categories of machine learning : supervised, unsupervised, and reinforcement learning. It highlights the evolution of machine learning techniques from the traditional statistical techniques such as linear regression and logistic regression and highlights the need for the adoption of neural networks. Additionally the paper explorers neural networks and their training in detailed steps. Every step such as forward and backward passes is explained with a focus on backpropagation.

1 Introduction

For centuries, humans have sought to predict the outcomes of various phenomena observed in the natural and social world. To aid in this endeavor, mathematics has served as a critical tool, employing methods such as linear regression and logistic regression. However, these traditional statistical techniques have limitations, particularly when handling complex, non-linear relationships or large datasets. In recent years, machine learning has emerged as a powerful extension, addressing these limitations by leveraging algorithms capable of learning patterns and making predictions with greater accuracy and flexibility.

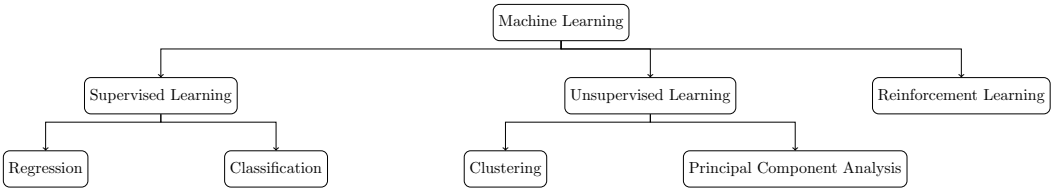
Machine Learning: Machine Learning is a method of artificial intelligence that allows machines to learn from the data provided to them. Machine Learning can be split into three different types:

- *Supervised Machine Learning :* Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial

intelligence. It is defined by its use of labeled data sets to train algorithms that to classify data or predict outcomes accurately. It includes techniques like regression and classification using neural networks. This is often taken as a curve fitting problem of a multi-dimensional function. This approach of supervised machine learning has existed for labelled data even before the introduction of neural networks.

- *Unsupervised Machine Learning :* Unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data.

- *Reinforcement Learning :* Reinforcement Learning (RL) is a branch of machine learning focused on making decisions to maximize cumulative rewards in a given situation. Unlike supervised learning, which relies on a training dataset with predefined answers, RL involves learning through experience. In RL, an agent learns to achieve a goal in an uncertain, potentially complex environment by performing actions and receiving feedback through rewards or penalties.



The concept of neural networks was first introduced in the paper about perceptron [2] in the year 1958 but only received small amounts of success due to the limitations in the computational capacity of the processors at that time. This led to the age that was called the AI winter.

In the 2000 when the computational power caught up with the requirements of these models we saw a surge in the usage of this. The paper AlextNext [1] was the first implementation of Convolution Neural Network that brought light to it.

Today, neural networks are one of the most adopted paradigms in computer science. This paper discuses the limitations of some of the most well-known techniques and there limitations that led to the adoption of neural networks in the sub domain of supervised machine learning.

2 Mathematical Approach

As discussed in the description of supervised machine learning, labelled data has been treated as a problem of curve fitting even before the introduction of

neural networks. There are multiple techniques used for curve fitting. Two of the most widely used to this day are - : *Linear Regression* for continuous value prediction and *Logistic Regression* for discrete value prediction.

2.1 Linear Regression

Regression is the statistical approach used to analyze the relationship between dependent variables and independent variables. The objective of this is to determine the most suitable function that fits the dataset. Linear regression uses linear function to perform this fitting task.

Mathematically,

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

where:

- y is the dependent variable (target).
- $x_1, x_2, \ldots, x_n$ are the independent variables (features).
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients that represent the weight of each feature.
- ϵ is the error term, capturing the difference between the predicted and actual values.

The objective in linear regression is to find the optimal values for $\beta_0, \beta_1, \ldots, \beta_n$ that minimize the sum of squared errors:

$$J(\beta_0, \beta_1, \ldots, \beta_n) = \frac{1}{2m} \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$$

where:

- m is the number of training examples.
- $y^{(i)}$ is the actual value for the i-th example.
- $\hat{y}^{(i)}$ is the predicted value for the i-th example.

The solution for β values can be obtained using gradient descent or the normal equation.

2.2 Logistic Regression

Logistic regression is used for binary classification tasks, where the goal is to predict one of two possible classes. Unlike linear regression, which predicts continuous values, logistic regression predicts the probability that a given input belongs to a particular class. The logistic function (sigmoid) is used to map predictions to probabilities.

The logistic regression model is given by:

$$p(y = 1|x) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)$$

where:

- $p(y = 1|x)$ is the probability of the output being 1 (class 1).
- $\sigma(z)$ is the logistic (sigmoid) function:

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

- $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients as in linear regression.

The loss function used in logistic regression is the *logistic loss* or *cross-entropy loss*:

$$L(\beta_0, \beta_1, \ldots, \beta_n) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(p^{(i)}) + (1 - y^{(i)}) \log(1 - p^{(i)}) \right]$$

where:

- $p^{(i)}$ is the predicted probability for the i-th example.
- $y^{(i)}$ is the actual class label (0 or 1).

The parameters $\beta_0, \beta_1, \ldots, \beta_n$ are learned by minimizing the loss function using optimization techniques like gradient descent.

In both regression and classification problems, there can be multiple different types of loss functions that can be used like *cross-entropy loss*, *maximum likelihood method*, *gradient descent* etc. The choice of the loss function will depend upon the data and architecture being used.

3 Limitations of Pure Statistical Approaches

Statistical approaches although being useful, evidently has limitations to them. Some of the limitations are -

- Assumption of linear relationships between the dependent and independent variables. This does not hold true in the real world. These methods used linear functions to reduce the complexity of modeling the functional and were not good at modeling non-linear representations. Example : *Stock Market*, *Image Recognition*, *Generative Models*, etc.

- Large complexity in modeling the data. A possible approach towards dealing with the lack of non-linear terms was to introduce higher-degree polynomials or more complex feature engineering, but this increases the dimensionality and can lead to overfitting.

- Inability to handle large and complex datasets efficiently. Statistical methods tend to struggle when data becomes high-dimensional, or when there is a large volume of data. This limitation was evident in tasks such as image recognition or speech recognition, where the complexity of the problem outstrips the capabilities of traditional statistical techniques.

- Overfitting and underfitting. Statistical methods like linear regression can easily overfit the training data if there are too many features, or underfit if the model is too simple. This is especially problematic in real-world applications where data can be noisy, sparse, or contain hidden patterns.

- Difficulty in capturing hierarchical and temporal dependencies. Many statistical models are not capable of effectively handling sequential data or hierarchical relationships (such as those found in text, speech, or image data). This is where neural networks, particularly deep learning models, come into play, offering more flexibility and the ability to learn complex relationships in data.

As these limitations became apparent, the need for more advanced techniques led to the development and widespread adoption of neural networks. Neural networks are capable of learning more complex, non-linear relationships between inputs and outputs, and are thus able to overcome many of the challenges posed by traditional statistical methods. This can be done via the use of multilayered architecture and activation functions.

Activation functions in particular are very useful when dealing with the problem of linearity.

4 Neural Networks and Deep Learning

Neural Networks are a branch of machine learning. They expand upon the concept of perceptron. Their nature of producing an output via calculating the weighted input and passing it to the next node make them suitable for modeling hidden relationship and making the data non-linear.

Deep learning refers towards the multi-layered architecture used to extract the hidden relationships between the dependent and independent variables.

4.1 The Architecture of Neural Networks

A basic neural network consists of three main types of layers:

- *Input Layer:* The input layer receives the raw data and passes it onto the subsequent layers.
- *Hidden Layers:* The hidden layers are where the computation happens. Each hidden layer consists of several neurons that apply activation functions to the weighted sum of inputs.
- *Output Layer:* The output layer produces the final prediction or classification based on the outputs of the hidden layers.

Unlike traditional statistical models, neural networks with the help of the multi-layered architecture are able to learn increasingly abstract representations of data with every layer. Due to this nature, we can model tasks like object recognition, speech processing, text generation with the help of neural networks with much ease.

4.2 Training Neural Networks

Training a neural networks is done in multiple individual steps. There will be link to the repository that uses the knowledge written in this paper to train a neural network on the MNIST dataset and distinguish the digits individually. Code at https://github.com/Ma-gi-cian/Machine-Learning-Paper The steps to train a neural network are :-

4.2.1 Forward Pass

The forward pass is the first step in the process of training a neural network. In this step we initialize the weights and biases for each neuron in the hidden layers and the output layers. A weighted sun of the inputs is taken followed by an activation function.

The activation function can be of different types like *relu*, *sigmoid*, *tanh*, etc. The choice of the activation function will depend upon the nature of the data. As the name suggest these are activation functions, that help in modeling complex data via activating and deactivating neurons that are deemed to be necessary for a particular data.

The input layer does not generally have any weights attached to it and just passess the data to the hidden layers.

For the l-th hidden layer, where $l = 1, 2, \ldots, L-1$:

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$

The above step produces the weighted sum which is further passed into an activation function.

$$a^{[l]} = g^{[l]}(z^{[l]})$$

This continues for all the hidden layers the only different comes into the last output layer. The activation function of the output layer depends upon purpose of the neural network.

The different types of activation functions that can be used are -
For regression (no activation or linear output):

$$\hat{y} = z^{[L]}$$

For binary classification (Sigmoid activation) :

$$\hat{y} = \sigma(z^{[L]}) = \frac{1}{1 + e^{-z^{[L]}}}$$

Binary classification means only two output like categorizing if the output has only two classes like 0 and 1 or true and false.
For multi-class classification (Softmax activation):

$$\hat{y}_i = \frac{e^{z_i^{[L]}}}{\sum_{j=1}^{n_L} e^{z_j^{[L]}}}, \quad \text{for } i = 1, 2, \ldots, n_L$$

Multi-class classification is used in problems where multiple classes are present. The problem of detecting the digit from the list of [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] or classification of a text into different emotions of sad, happy, angry, etc.

Here, $\hat{y}$ represents the final output of the neural network.

4.2.2 Backward Pass (Backpropagation[3][4])

The backward pass, that uses the technique of backpropagation, is the process of updating the weights and biases of a neural network by calculating the gradient of the loss function with respect to each parameter. This process ensures that the network minimizes the error during training.

Even before we begin to update the weights, we need to determine the amount of error that exists between the actual output and the output generated via forward pass. This is done via the *loss function*.

Loss Function To calculate the gradients, we first define the loss function J. For instance, in binary classification:

$$J(W, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

where:

- m is the number of training examples.
- $y^{(i)}$ is the actual label.
- $\hat{y}^{(i)}$ is the predicted output.

Gradients for Output Layer For the output layer L:

$$\delta^{[L]} = \hat{y} - y$$

$$\frac{\partial J}{\partial W^{[L]}} = \frac{1}{m}\delta^{[L]}(a^{[L-1]})^T$$

$$\frac{\partial J}{\partial b^{[L]}} = \frac{1}{m}\sum_{i=1}^{m}\delta_i^{[L]}$$

Gradients for Hidden Layers For each hidden layer $l = L-1, L-2, \ldots, 1$:

$$\delta^{[l]} = \left(W^{[l+1]T}\delta^{[l+1]}\right) \odot g'^{[l]}(z^{[l]})$$

$$\frac{\partial J}{\partial W^{[l]}} = \frac{1}{m}\delta^{[l]}(a^{[l-1]})^T$$

$$\frac{\partial J}{\partial b^{[l]}} = \frac{1}{m}\sum_{i=1}^{m}\delta_i^{[l]}$$

Update Parameters Using gradient descent, update weights and biases:

$$W^{[l]} := W^{[l]} - \alpha\frac{\partial J}{\partial W^{[l]}}$$

$$b^{[l]} := b^{[l]} - \alpha\frac{\partial J}{\partial b^{[l]}}$$

where α is the learning rate.

Summary of Backpropagation For each layer l:

- Compute $\delta^{[l]}$ (error term for layer l).
- Calculate gradients $\frac{\partial J}{\partial W^{[l]}}$ and $\frac{\partial J}{\partial b^{[l]}}$.
- Update parameters using gradient descent.

4.3 Evaluation of the Model

After training the neural network, it is essential to evaluate its performance using a test dataset that was not seen during the training process. This evaluation helps in assessing how well the model generalizes to unseen data.

4.3.1 Accuracy Calculation

Generally when we talk about evaluating a model we tend to talk about the accuracy of the model, which is the amount of correct predictions that the model made when compared to the actual prediction.

Evaluation for Classification : The primary metric used for evaluating a model is often accuracy, which measures the proportion of correct predictions made by the model. For classification problems, accuracy is defined as:

$$\text{Accuracy} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1} \left(\hat{y}^{(i)} = y^{(i)} \right)$$

Where:

- m is the number of test examples.
- $\mathbf{1}$ is the indicator function that returns 1 if $\hat{y}^{(i)} = y^{(i)}$, meaning the prediction is correct, and 0 otherwise.
- $\hat{y}^{(i)}$ is the predicted output for the i-th example.
- $y^{(i)}$ is the actual label for the i-th example.

In the case of multi-class classification, accuracy is still calculated in the same way, but for each class, the model's prediction is compared to the true class.

Evaluation for Regression In regression problems, accuracy is not typically used as a metric. Instead, other metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are commonly employed to evaluate the model's performance.

Mean Absolute Error (MAE) The Mean Absolute Error (MAE) measures the average of the absolute differences between predicted values and actual values:

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^{m} \left| \hat{y}^{(i)} - y^{(i)} \right|$$

Where:

- $\hat{y}^{(i)}$ is the predicted value for the i-th example.
- $y^{(i)}$ is the actual value for the i-th example.
- m is the number of test examples.

Mean Squared Error (MSE) The Mean Squared Error (MSE) measures the average of the squared differences between predicted and actual values. It penalizes larger errors more than MAE:

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

Root Mean Squared Error (RMSE) The Root Mean Squared Error (RMSE) is the square root of the MSE and provides the error in the same units as the target variable:

$$\text{RMSE} = \sqrt{\text{MSE}}$$

These metrics are useful in assessing the performance of regression models by measuring the average error between the predicted and actual values.

4.3.2 Confusion Matrix

For multi-class problems we can also take the help of a confusion matrix. The confusion matrix is an $n \times n$ matrix where n is the number of classes. Each element C_{ij} represents the number of times class i was predicted as class j.

$$C_{ij} = \text{Number of times class } i \text{ was predicted as class } j$$

From the confusion matrix, various other metrics can be derived, such as:

- *Precision* : The fraction of relevant instances among the retrieved instances.

  $$\text{Precision} = \frac{TP}{TP + FP}$$

  Where: - TP is the number of true positives (correct predictions for the class). - FP is the number of false positives (incorrect predictions where the class was incorrectly predicted).

- *Recall* : The fraction of relevant instances that have been retrieved.

  $$\text{Recall} = \frac{TP}{TP + FN}$$

  Where: - FN is the number of false negatives (instances that were incorrectly classified as other classes).

- *F1 Score* : The harmonic mean of precision and recall. The significance of the F1 Score is that it combines both precision and recall to take into account both the values while evaluating the model

  $$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.3.3 Loss Evaluation

In addition to accuracy, the loss function can be used to evaluate how well the model is performing. The loss should be minimized during training, and the evaluation of the loss on the test dataset helps to identify if the model has overfitted to the training data. If the loss on the test set is much higher than on the training set, it may indicate overfitting.

For classification, the cross-entropy loss is often used as the loss function. The evaluation of loss on the test set can be done as follows:

$$L_{\text{test}} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

Where L_{test} is the average loss on the test set.

4.3.4 Model Generalization

Evaluating a model's ability to generalize is crucial. If a model performs well on the training data but poorly on the test data, it is a sign of overfitting, meaning the model has memorized the training data but cannot generalize to unseen data. Regularization techniques, such as dropout or L2 regularization, can help prevent overfitting and improve generalization.

4.3.5 Cross-Validation

Cross-validation is a technique used to assess the generalization ability of the model by splitting the dataset into multiple subsets (folds). The model is trained on some folds and tested on the remaining fold. This process is repeated for each fold, and the results are averaged to provide a more robust estimate of model performance. Cross-validation is particularly useful when the dataset is small and can help detect if the model is overfitting.

5 Conclusion

Machine learning has been extremely helpful step towards data prediction. Although statistical learning methods like linear regression and logistic regression are helpful and still useful for small datasets that are not too complex, evidently have limitations. These limitations can be comfortably overcome via the use of neural networks. Neural networks with their ability to learn abstract hierarchical representations of data have been helpful in the field of GANs (Generative models), speech recognition, etc. Neural networks may appear to be mysterious and a black box but are only essentially curve fitting tools better than what we used previously. As computational power continues to grow, the role of machine learning in solving complex, real-world problems will only increase.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[2] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[3] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[4] Paul Werbos. *Beyond the Hidden Layer: New Methods for Training Neural Networks*. PhD thesis, Harvard University, Cambridge, MA, 1974.