HIS PROJECT - TSA

Task 05

Author
Alberto Mejia

Contents

1 Chapter 11: Introduction to Deep Learning

1.1 What is deep learning and why now?

- **Increase in compute availability**
- **Increase in data availability**

1.2 Perceptron - The first neural network

The Perceptron, developed by Frank Rosenblatt in the 1950s, is one of the first neural network models. It functions by taking weighted inputs, summing them, and then outputting a binary result based on a threshold. This model mimics the basic function of a biological neuron but is much simpler. Despite its simplicity and limitations (such as handling only linearly separable problems), the Perceptron laid the groundwork for more complex neural networks.

- **Inputs:** These are the real-valued inputs that are fed to a Perceptron. This is like the dendrites in neurons that collect the input.
- **Weighted sum:** Each input is multiplied by a corresponding weight and summed up. The weights determine the importance of each input in determining the outcome.
- **Non-Linearity:** The weighted sum goes through a non-linear function.Real-world data is often complex and not linearly separable. A nonlinear function allows the neural network to capture and model these nonlinear relationships in the data.

1.3 Components of a deep learning system

Deep learning is presented as a highly modular system, characterized by two key properties of its parametrized modules:

- **Output production:** These modules can generate an output from a given input through a series of computational steps.
- **Feedback adjustment:** When provided with the desired output, the modules can communicate back to their inputs, indicating how they should change to more closely align with this desired outcome. This feedback mechanism helps in iteratively adjusting the inputs to reduce the difference between the actual and desired outputs.

The core of this concept is the use of differentiable functions, which is crucial for gradient-based optimization methods commonly employed in deep learning. This differentiability allows for efficient computation of gradients, which are used to update the parameters of the model in a direction that minimizes the error or loss. This makes deep learning models adaptable and effective for a wide range of complex tasks.

1.4 Linear layers and activation functions

Learns the best features by which we can make the problem linearly separable. Linearly separable means when we can separate the different classes (in a classification problem) with a straight line.

1.4.1 Linear Transformations (In hidden layers)

- **Purpose:** Linear transformations in neural networks are typically accomplished through operations like weighted sums. They are used to project input data into a different space.
- **Function:** A linear transformation combines input features in a linear way (e.g., summing up inputs each multiplied by their respective weights). This process is essential for feature combination and transformation.
- **Implementation and Impact:** Linear transformations alone cannot capture complex patterns in data, especially if the data is not linearly separable. They are limited to linear relationships.

1.4.2 Intermediate Activation Functions (In hidden layers)

- **Purpose:** Activation functions introduce nonlinearity into the network. They help the network learn and represent more complex relationships in the data.
- **Function:** After the linear transformation, the activation function processes the result to produce a nonlinear output. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, and tanh.
- **Implementation and Impact:** Without activation functions, a neural network, regardless of how many layers it has, would essentially be a linear regression model, unable to solve complex problems like image recognition or natural language processing.

Linear transformations handle the combination and re-scaling of input features, while activation functions introduce the necessary nonlinearity that allows the network to model complex patterns. A typical layer in a neural network sequentially applies a linear transformation followed by a nonlinear activation function. This combination enables the network to learn from and make predictions on complex, real-world data.

1.4.3 Output Activation Functions (In output layer)

- **Purpose:** The output activation function is critical in adapting the final layer's output of a neural network to the format required for a specific task, such as regression, binary classification, or multiclass classification. It ensures that the network's output is interpretable and suitable for the problem at hand.

- **Function:** This function processes the output of the network's final linear transformation to produce a result that aligns with the nature of the target variable. For instance, in a binary classification task, it converts the final layer's output into a probability (between 0 and 1), while in a regression task, it might leave the output as is (linear activation) or apply a transformation to ensure the output falls within a certain range.

- **Implementation and Impact:** The selection and application of an output activation function have profound implications on the training dynamics, model accuracy, and result interpretability. It needs to be compatible with the loss function used, directly influencing how the model's performance is evaluated and optimized during training. This choice also determines how the final output is presented, making it crucial for ensuring the model's outputs are relevant and comprehensible for the given task.

1.4.4 Loss function

- **Function:** A loss function, also known as a cost function, quantitatively measures how far the predictions of a neural network are from the actual target values. In mathematical terms, it calculates the error or difference between the predicted output and the true output. The specific form of a loss function can vary depending on the type of task. For example, Mean Squared Error (MSE) is common in regression tasks, while Cross-Entropy loss is often used in classification tasks.

- **Purpose:** The primary purpose of a loss function is to guide the training process of the neural network. By providing a quantifiable metric of how well the model is performing, the loss function becomes the objective that the training process aims to minimize. It serves as a feedback mechanism for the model, indicating how far off its predictions are, and thus how it needs to adjust its weights during training.

- **Implementation and Impact:** During training, the neural network uses an optimization algorithm, typically some form of gradient descent, to adjust its weights in a way that minimizes the loss function. The loss function must be compatible with the output activation function of the neural network. For instance, using a softmax activation in the output layer typically pairs with a cross-entropy loss function.

1.5 Gradient descent

1.5.1 Forward Computation (Forward Propagation)

- This involves passing input data through the network, layer by layer, using the computations defined in each layer, to produce an output.

- The output is then compared to the desired output using a loss function to determine how close the model's prediction is to the actual target.

1.5.2 Backward Computation (Backward Propagation)

- Once the output is obtained and the loss is calculated, this step involves computing the gradient of the loss function with respect to all the parameters in the network.
- The gradient, a generalization of derivatives for multivariate functions, indicates the rate of change of the loss with respect to changes in the network's parameters.

1.5.3 Gradient and its Importance

- Gradients, akin to slopes in high school mathematics, inform us about the local slope of a function.
- By understanding the gradient of the loss function, gradient descent, a mathematical optimization technique, can be employed to adjust the network's parameters in a way that minimizes the loss, thus optimizing the model's performance.