мишек ликола

5130203/20101

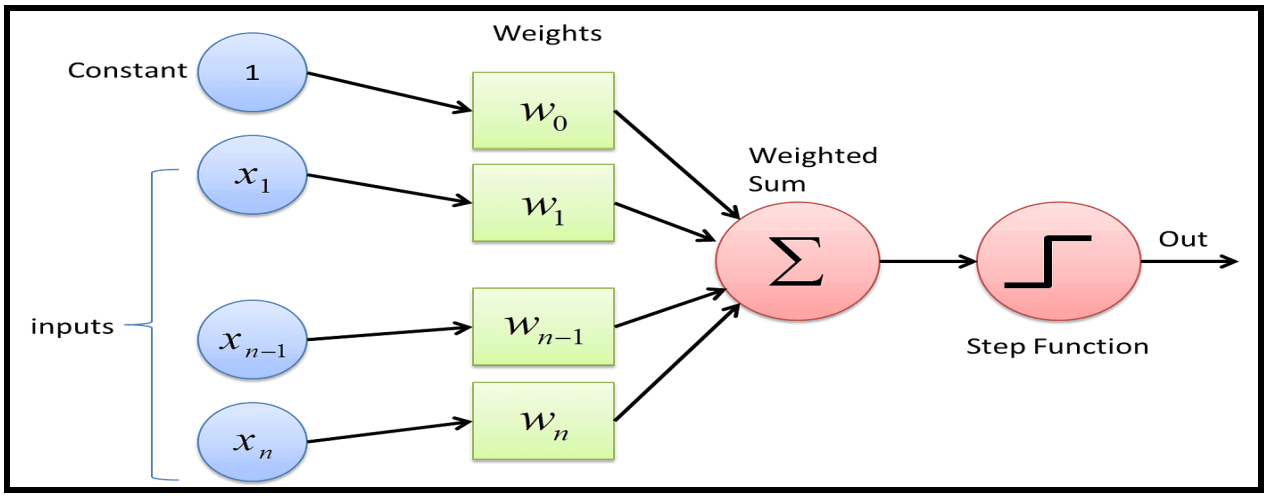
1. Perceptron

1. Model Architecture:

The Perceptron consists of:

- **Input layer**: The input vector $\mathbf{x} = [x_1, x_2, ..., x_n]$ represents the features of the dataset.
- **Weights**: The model has a set of weights $\mathbf{w} = [w_1, w_2, ..., w_n]$, each associated with a feature.
- **Bias**: The bias b allows the model to adjust the decision boundary.
- **Output**: The output $\hat{y} \in \{0, 1\}$, representing the predicted class label.

The structure can be visualized as:



2. Vector Representation

- **Inputs**: The inputs are represented as a column vector $\mathbf{X}$, which consists of $n+1$ elements, including a bias term. The bias is set to 1 for all input vectors:

$$\mathbf{X} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad x_i \in \mathbb{R},\ \forall i = 1, \ldots, n.$$

  Here, $x_1, x_2, \ldots, x_n$ are the feature values of the input.

- **Weights**: The weights are represented as a column vector $\mathbf{W}$, which has $n+1$ elements corresponding to each input (including the bias weight):

$$\mathbf{W} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}, \quad w_i \in \mathbb{R},\ \forall i = 0, \ldots, n.$$

- **Outputs**:
  - $\hat{y}$ is the predicted output, which is a binary value ($\hat{y} \in \{0, 1\}$).
  - y is the actual (ground truth) label.
- **Linear Combination**: Using these vectors, the weighted sum z is computed as:

$$z = \mathbf{X}^T \mathbf{W},$$

3. Mathematical Formulation

- **Linear Combination**:
  The perceptron computes a weighted sum of the inputs, including the bias term. This is represented as:

$$z = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \mathbf{W}^T\mathbf{X},$$

  where $\mathbf{W}$ is the weight vector, $\mathbf{X}$ is the input vector, and z is the weighted sum.
- **Activation Function**:
  The perceptron uses a step activation function (Φ), defined as:

$$\Phi(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ 0 & \text{if } z < 0. \end{cases}$$

- **Loss Function**:
  In the perceptron algorithm, the loss function is defined to penalize misclassified examples. A common formulation is:

$$L = \max(0, -y \cdot z),$$

  where y is the true label ($y \in \{-1, 1\}$) and $z = \mathbf{W}^T\mathbf{X}$.

4. Prediction Calculation

The Perceptron model predicts an output ($\hat{y}$) by applying the activation function to the weighted sum of inputs:

$$\hat{y} = \Phi(z) = \Phi(\mathbf{W}^T\mathbf{X}),$$

where $z = \mathbf{W}^T\mathbf{X}$ represents the linear combination of weights and inputs, and $\Phi(z)$ determines the classification output ($\hat{y} = 1$ or $\hat{y} = 0$) based on the step function.

---

5. Gradient Descent Algorithm

In classical Perceptron learning, gradient descent is not explicitly used. Instead, a rule-based approach is employed to correct errors for misclassified examples. The algorithm adjusts the weights and bias incrementally to reduce classification errors.

---

6. Weight and Bias Update Formulas

When a misclassification occurs, the model updates weights and bias iteratively as follows:

- **Weight Update**:

$$w_i^{(t+1)} = w_i^{(t)} + \eta \cdot (y - \hat{y}) \cdot x_i,$$

where:

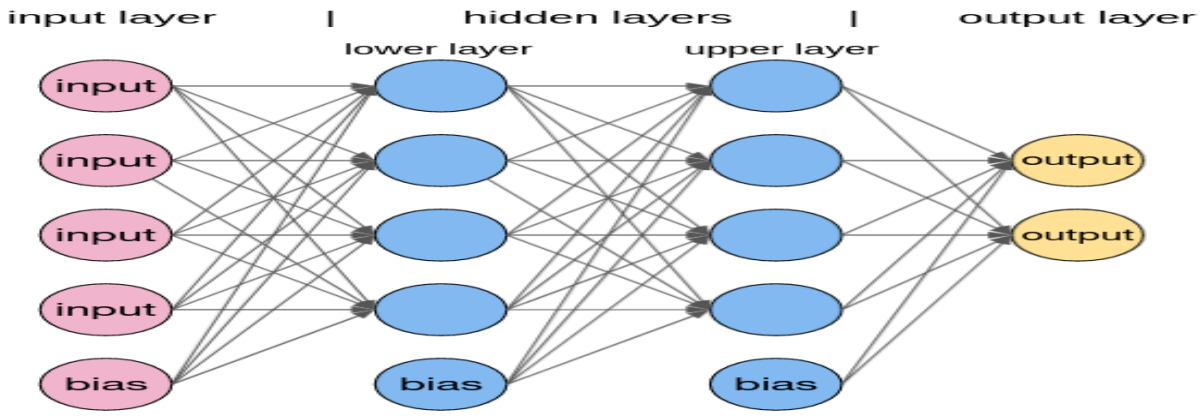
- $w_i^{(t)}$ is the weight at the current iteration.
- η is the learning rate, controlling step size.
- y is the actual output.
- $\hat{y}$ is the predicted output.
- x_i is the corresponding input feature.
- **Bias Update**:

$$b^{(t+1)} = b^{(t)} + \eta \cdot (y - \hat{y}),$$

where b represents the bias term.

2 .Multilayer Perceptron

Model Architecture:



2. Vector Representation of Data

The inputs, hidden layer, and outputs can be expressed as vectors for clarity:

1. **Input vector X:**

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

2. **Hidden layer H:**

$$H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_m \end{bmatrix}$$

3. **Output vector Y:**

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_b \end{bmatrix}$$

Weights and Biases:

- Weights between input and hidden layers:
  $W^h \in \mathbb{R}^{m \times n}$,
  where W^h_{ik} represents the weight connecting the k-th input to the i-th hidden node.
- Weights between hidden and output layers:
  $W^o \in \mathbb{R}^{b \times m}$.
- Bias terms for hidden and output layers:
  $\theta^h \in \mathbb{R}^m$, $\theta^o \in \mathbb{R}^b$.

3. Mathematical Formulation

1. **Linear Combination:**
   For the hidden layer:

$$net_i^h = \sum_{k=1}^{n} (x_k \cdot W_{ik}^h) + \theta_i^h$$

   For the output layer:

$$net_j^o = \sum_{i=1}^{m} (f_i^h \cdot W_{ji}^o) + \theta_j^o$$

2. **Activation Function:**
   Sigmoid function for hidden and output layers:

$$f_i^h = \Phi(net_i^h) = \frac{1}{1 + e^{-net_i^h}}, \quad f_j^o = \Phi(net_j^o) = \frac{1}{1 + e^{-net_j^o}}$$

3. **Loss Function:**
   Mean Squared Error (MSE):

$$L = \frac{1}{2} \sum_{j=1}^{b} (y_j - \hat{y}_j)^2$$

4. Predictions

The predicted value $\hat{y}_j$ is computed as:

$$\hat{y}_j = \Phi(net_j^o)$$

---

5. Gradient Descent Algorithm

The backpropagation algorithm is used to update weights and biases iteratively. The gradients are computed by propagating the errors backward through the network.

6. Update Rules for Weights and Biases

Output Layer Updates:

- Weights:

$$W_{ji}^{o}(t+1) = W_{ji}^{o}(t) - \mu \frac{\partial L}{\partial W_{ji}^{o}}$$

- Biases:

$$\theta_{j}^{o}(t+1) = \theta_{j}^{o}(t) - \mu \frac{\partial L}{\partial \theta_{j}^{o}}$$

Hidden Layer Updates:

- Weights:

$$W_{ik}^{h}(t+1) = W_{ik}^{h}(t) - \mu \frac{\partial L}{\partial W_{ik}^{h}}$$

- Biases:

$$\theta_{i}^{h}(t+1) = \theta_{i}^{h}(t) - \mu \frac{\partial L}{\partial \theta_{i}^{h}}$$

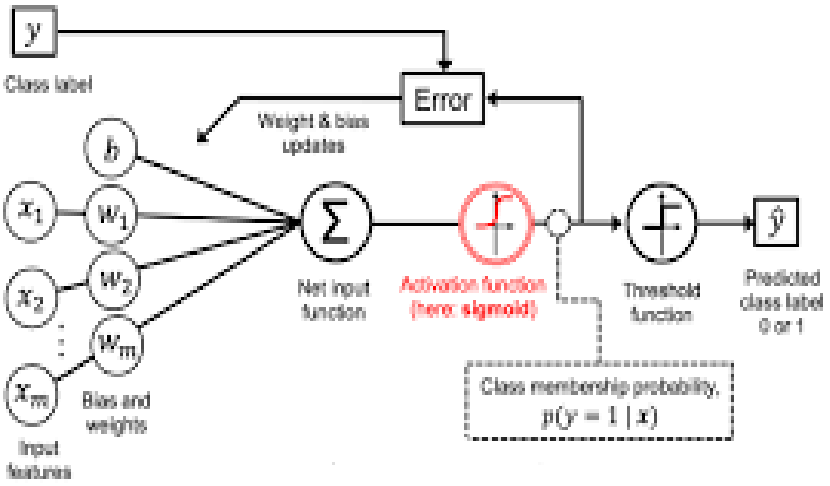
Gradients for the Hidden Layer:

$$\delta_{i}^{h} = \Phi'(net_{i}^{h}) \cdot \sum_{j=1}^{b} (\delta_{j}^{o} \cdot W_{ji}^{o})$$

Here, $\delta_{j}^{o} = -(y_{j} - \hat{y}_{j}) \cdot \Phi'(net_{j}^{o})$ is the output layer error.

3.Logistic Regression

Model Architecture:



Logistic Regression: Key Concepts and Formulations

1. Vector Representation of Data

- **Input Vector**:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

  Represents the features of the data points.

- **Target Output**:
  $y \in \{0, 1\}$, representing binary class labels.

2. Mathematical Formulations

- **Linear Combination**:
  Compute the weighted sum of inputs and bias:

$$z = \mathbf{w}^T \mathbf{x} + b$$

  Where:

  - $\mathbf{w} \in \mathbb{R}^n$ is the weight vector.
  - $b \in \mathbb{R}$ is the bias term.

- **Activation Function (Sigmoid)**:
  Transform z into a probability using the sigmoid function:

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

  $\hat{y} \in (0, 1)$ represents the predicted probability of $y = 1$.

- **Loss Function (Binary Cross-Entropy)**:
  To quantify the difference between predicted and true outputs:

$$L(\mathbf{w}, b) = -\Big[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\Big]$$

3. Prediction Calculation

- Logistic Regression predicts the probability of $y = 1$:

$$\hat{y} = \sigma(\mathbf{w}^T \mathbf{x} + b)$$

- Decision Rule for Classification:

$$y = \begin{cases} 1 & \text{if } \hat{y} \geq 0.5, \\ 0 & \text{if } \hat{y} < 0.5. \end{cases}$$

---

4. Gradient Descent Algorithm

- **Objective**: Minimize the loss function $L(\mathbf{w}, b)$ by iteratively updating the weights and bias.
- **Gradients**:
  Partial derivatives of the loss function w.r.t. weights and bias:

$$\nabla_{\mathbf{w}} L = (\hat{y} - y)\mathbf{x}, \quad \nabla_b L = (\hat{y} - y)$$

- **Update Rules**:
  Using the learning rate $\eta > 0$:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L$$

$$b \leftarrow b - \eta \nabla_b L$$