Assignment 1 - Multiclass Logistic Regression

Advanced Topics in Neural Networks

03 October 2023

1 Logistic Regression

1.1 Introduction to Logistic Regression

Logistic Regression is a statistical method for modeling binary outcomes. In the context of machine learning, it is used for binary classification tasks, where the goal is to categorize instances into one of two classes. Logistic Regression can be extended to multiclass classification.

Here, we introduce the core components involved in Logistic Regression:

- $\mathbf{x}$: The input feature vector containing the attributes of the instance being classified.
- $\mathbf{w}$: The weight vector representing the coefficients that are learned during the training process. Each element of $\mathbf{w}$ corresponds to a feature in $\mathbf{x}$.
- b: The bias term, which adjusts the decision boundary away from the origin, increasing the model's flexibility.
- z: The linear combination of the weights, the feature vector, and the bias, calculated as $z = \mathbf{w}^\top \mathbf{x} + b$.
- $\sigma(z)$: The sigmoid function applied to z to squash the output between 0 and 1, providing a probability score for the positive class.

Notation Legend:

- $\frac{\delta L}{\delta w}$ and $\frac{\delta L}{\delta b}$: Represent the partial derivatives of the loss function L with respect to the weights $\mathbf{w}$ and bias b, respectively.
- $\nabla_{\mathbf{w}} L$ and $\nabla_b L$: Are alternative notations for the gradients of the loss function with respect to the weights and bias. You will also see them noted as simply Δw and Δb.
- $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L$: Indicates the update rule for the weights, where η is the learning rate and $\nabla_{\mathbf{w}} L$ is the gradient of the loss with respect to the weights.

In the following subsections, we delve into the mathematical formulations and procedures for binary and multiclass Logistic Regression, accompanied by an example to illustrate the computations.

1.2 Binary Logistic Regression

We begin by computing the linear combination of the input features and weights, adding the bias term.

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

We then apply the sigmoid function to transform the linear combination into a probability.

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

The cross-entropy loss function measures the dissimilarity between the predicted probability and the actual class label.

$$L(\hat{y}, y) = -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right)$$

We calculate the gradient of the loss with respect to the weights.

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

Similarly, we compute the gradient of the loss with respect to the bias.

$\nabla_b L = \hat{y} - y$

The weights are updated by subtracting a fraction of the gradient, scaled by the learning rate η.

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

The bias is updated in a similar manner.

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

1.3 Example

Given data:
Using the given data, we will demonstrate the steps to compute the predicted output.

$\mathbf{x} = [1, 3, 0]$
$\mathbf{w} = [-0.6, -0.5, 2]$
$b = 0.1$
$y = 1$

Compute the weighted sum z and the predicted probability $\hat{y}$.

$$\begin{aligned} z &= \mathbf{w}^\top \mathbf{x} + b \\ &= (-0.6 \times 1) + (-0.5 \times 3) + (2 \times 0) + 0.1 \\ &\approx -2.0 \\ \hat{y} &= \sigma(z) \\ &= \frac{1}{1 + \exp(2.0)} \\ &\approx 0.12 \end{aligned}$$

Compute the cross-entropy loss.

$$\begin{aligned} L(\hat{y}, y) &= -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right) \\ &= -\log(0.12) \\ &\approx 2.12 \end{aligned}$$

Compute the gradients for weights and bias.

$$\begin{aligned} \nabla_{\mathbf{w}} L &= (0.12 - 1) \cdot [1, 3, 0] \\ &= [-0.88, -2.64, 0] \\ \nabla_b L &= 0.12 - 1 \\ &\approx -0.88 \end{aligned}$$

These gradients are used to update the model parameters, leading to a better fit to the training data.

1.4 Multiclass Logistic Regression

For multiple classes, we compute a set of linear combinations for each class.

$\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$

The softmax function is applied to transform the linear combinations into class probabilities.

$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$

The resulting vector $\hat{\mathbf{y}}$ contains the predicted probabilities for each class.

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}, \quad i = 0, \ldots, n$$

The loss and gradient calculations, as well as parameter updates, follow the general pattern outlined in the binary case but are extended to accommodate multiple classes.

2 Exercise

Due date: End-of-Day 09.10.2023

Computing the Parameter Update
Your task is to perform one update step of the weights and biases using the given data and multiclass logistic regression. The provided data includes a feature vector $\mathbf{x}$, initial weights matrix $\mathbf{W}$, bias vector $\mathbf{b}$, and the true class labels in one-hot encoded format $\mathbf{y}$.

$\mathbf{x} = [1, 3, 0]$,
$\mathbf{W} = \begin{bmatrix} 0.3 & 0.1 & -2 \\ -0.6 & -0.5 & 2 \\ -1 & -0.5 & 0.1 \end{bmatrix}$,
$\mathbf{b} = [0.1, 0.1, 0.1]$,
$\mathbf{y} = [0, 1, 0]$.

Begin by computing the linear combinations $\mathbf{z}$ for each class:

$\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$.

Apply the softmax function to get the predicted probabilities $\hat{\mathbf{y}}$:

$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z})$.

Compute the gradient of the loss with respect to $\mathbf{z}$ using the cross-entropy loss and the true labels $\mathbf{y}$:

$\nabla_{\mathbf{z}} L = \hat{\mathbf{y}} - \mathbf{y}$.

Now, compute the gradients with respect to the weights $\mathbf{W}$ and biases $\mathbf{b}$:

$\nabla_{\mathbf{W}} L = \nabla_{\mathbf{z}} L \mathbf{x}^\top$,
$\nabla_{\mathbf{b}} L = \nabla_{\mathbf{z}} L$.

Finally, update the weights and biases using a learning rate η:

$\mathbf{W} \leftarrow \mathbf{W} - \eta \nabla_{\mathbf{W}} L$,
$\mathbf{b} \leftarrow \mathbf{b} - \eta \nabla_{\mathbf{b}} L$.

Fill in the intermediate and final values for these computations. Analyze how different elements contribute to the update and understand the effect of the learning rate η.

Obs. You can solve this exercise on paper or programmatically using Python with Numpy or PyTorch basic operators. Please upload your solutions on your personal fork of the **Template Repository** [1]

Extra: Use your implementation of Multiclass Logistic Regression to train a model on **this wine quality dataset** [2] to learn to classify the "quality" class.

[1] https://github.com/Tensor-Reloaded/Advanced-Topics-in-Neural-Networks-Template-2023/
[2] https://github.com/aniruddhachoudhury/Red-Wine-Quality/blob/master/winequality-red.csv