Binary Logistic Regression Worked Examples

Reading:
Logistic Regression Materials
Patrick Loeber video / code

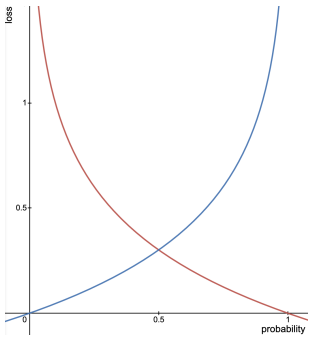
Logistic Regression

Logistic regression models are used for classification. Binomial (binary) models (discussed here) are trained to determine if a sample belongs to a class or not. Multinomial models classify a sample into one of many classes. Binary logistic regression models base the prediction (0 or 1) on the probability of the sample belonging to the class.

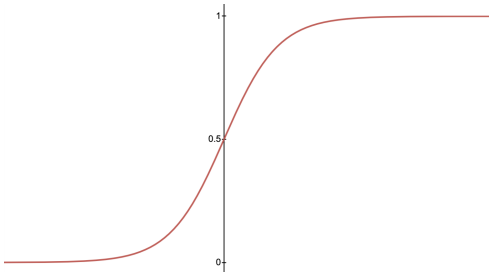
Logistic regression models are similar to linear regression models in that they both use **gradient descent** to minimize the **loss**, and a **learning rate** is used to control how much the weights and bias are updated at each epoch. However, in logistic regression the **cross-entropy loss** function is used instead of MSE, and the **sigmoid function** is applied to the linear output to get a probability.

Consider the cross entropy loss function, with loss plotted on the x-axis, and predicted probability on the x-axis. The blue curve represents the loss for samples whose gold value is 0, and the red curve represents the loss for samples whose gold value is 1. Notice that the loss is high if the prediction is wrong, and low if the prediction is correct.

The sigmoid function maps any value (on the x-axis) to a value between 0 and 1. We first calculate the linear output using the current weights and bias. Then we apply the sigmoid function to the linear output to calculate a probability. If the probability is greater than the decision boundary, the prediction is 1 (sample belongs to the class), otherwise the prediction is 0 (sample does not belong to the class).



(a) Cross Entropy Loss



(b) Sigmoid

1 Prediction

Prediction is done by applying the sigmoid function to the linear output, then using a decision boundary to determine the class (0 or 1 for binary models).

$$\text{linear\_output} = \boldsymbol{X} \cdot \boldsymbol{w} + b$$

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

$$Sigmoid(\text{linear\_output}) = \frac{1}{1 + e^{-\text{linear\_output}}}$$

$$Sigmoid(\text{linear\_output}) = \frac{1}{1 + e^{-(x \cdot w + b)}}$$

$$y\_pred = \begin{cases} 1 & \text{if } Sigmoid(\text{linear\_output}) \geq \text{boundary} \\ 0 & \text{otherwise} \end{cases}$$

1.1 Prediction Exercise

A logistic regression model was trained on a 3-feature dataset. The final weights and bias after training are:

$$\boldsymbol{w} = \begin{bmatrix} -4 \\ 2 \\ 3 \end{bmatrix} \qquad bias = 1$$

Calculate the predicted value of the following test data, which contains one sample. Use a decision boundary of 0.5.

$$\boldsymbol{X\_test} = \begin{bmatrix} 3 & -1 & 4 \end{bmatrix}$$

Solution:

$$\begin{aligned} linear\_output &= \boldsymbol{X} \cdot \boldsymbol{w} + b \\ &= \begin{bmatrix} 3 & -1 & 4 \end{bmatrix} \begin{bmatrix} -4 \\ 2 \\ 3 \end{bmatrix} + 1 \\ &= -2 + 1 \\ &= -1 \end{aligned}$$

$$\begin{aligned} y\_pred\_prob &= sigmoid(-1) \\ &= \frac{1}{1 + e^{-(-1)}} \\ &\approx \frac{1}{1 + 2.72} \\ &\approx .269 \end{aligned}$$

The y_pred_prob $<$ decision_boundary, so the prediction y_pred is 0.

2 Training

During training, feature weights and bias are updated to "fit" the training data. The optimal weights and bias usually can not be reached by processing the training data only once. The training data is processed over many iterations (called epochs). The loss should decrease at each epoch. The learning rate controls how much the weights and bias are adjusted at each epoch. If the learning rate is too high, the adjusted weights may cause a higher loss. If the learning rate is too low, the loss will not approach 0 in the set number of epochs. Training is the same as for linear regression, except that the sigmoid function is applied to the linear output in order to get probability predictions based on the current weights. Note that the derivative of the cross-entropy loss function is the same as that of the MSE cost function.

2.1 Training Exercise

Suppose you are training a binary logistic regression model (using a learning rate of .1) on the following 2-feature training data:

$$\boldsymbol{X\_train} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 2 & 3 \end{bmatrix} \qquad \boldsymbol{y\_train} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

At epoch x, the weights and bias are:

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad bias = -4$$

Calculate the weights and bias at epoch x+1 (i.e. after updating the weights and bias once).

Solution:

Calculate the linear outputs (one for each sample in the training data) with the current weights and bias:

$$linear\_model = \boldsymbol{X} \cdot \boldsymbol{w} + b$$
$$= \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + -4 = \begin{bmatrix} 5 \\ 11 \\ 8 \end{bmatrix} - 4 = \begin{bmatrix} 1 \\ 7 \\ 4 \end{bmatrix}$$

[Note the shapes of **X** (3,2) and **w** (2,1). The dot product results in a matrix of shape (3,1).]

Get probability predictions by applying the sigmoid function to the linear values:

$$y\_pred = sigmoid(linear\_model)$$
$$= sigmoid(\begin{bmatrix} 1 \\ 7 \\ 4 \end{bmatrix})$$
$$= \begin{bmatrix} \frac{1}{1+e^{-1}} \\ \frac{1}{1+e^{-7}} \\ \frac{1}{1+e^{-4}} \end{bmatrix}$$
$$\approx \begin{bmatrix} .731 \\ .999 \\ .982 \end{bmatrix}$$

Find the delta of the predictions and gold values. **Following the Patrick Loeber video, we use y_pred -y to avoid multiplication by -1 in the gradient calculations.**

$$y\_pred - y = \begin{bmatrix} .731 \\ .999 \\ .982 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -.269 \\ .999 \\ -.018 \end{bmatrix}$$

Calculate the gradients, which are the derivatives of the cost function wrt w, and wrt b. The derivatives for the cross-entropy loss function are the same as for MSE loss function. The derivatives are averaged over the samples for each weight. For $\frac{\partial f}{\partial w}$ we need transpose the training data to perform the dot product along the feature axis.

$$\frac{\partial f}{\partial w} = \frac{1}{n\_samples} * \boldsymbol{X}^T \cdot (y\_pred - y)$$
$$= (1/3) * \begin{bmatrix} 1 & 3 & 2 \\ 2 & 4 & 3 \end{bmatrix} \cdot \begin{bmatrix} -.269 \\ .999 \\ -.018 \end{bmatrix}$$
$$= (1/3) * \begin{bmatrix} 2.692 \\ 3.405 \end{bmatrix}$$
$$= \begin{bmatrix} .897 \\ 1.135 \end{bmatrix}$$

$$\frac{\partial f}{\partial b} = \frac{1}{n\_samples} * \Sigma(y\_pred - y)$$
$$= (1/3) * (-.269 + .999 + -.018)$$
$$= (1/3) * .712$$
$$= .237$$

Update the weights and bias using the learning rate:

$$\begin{aligned}\boldsymbol{w} &= \boldsymbol{w} - lr * \frac{\partial f}{\partial w} \\ &= \begin{bmatrix}1\\2\end{bmatrix} - .1 * \begin{bmatrix}.897\\1.135\end{bmatrix} \\ &= \begin{bmatrix}1\\2\end{bmatrix} - \begin{bmatrix}.0897\\.1135\end{bmatrix} \\ &= \begin{bmatrix}.91\\1.887\end{bmatrix}\end{aligned}$$

$$\begin{aligned}bias &= bias - lr * \frac{\partial f}{\partial b} \\ &= -4 - .1 * .237 \\ &= -4.024\end{aligned}$$