Logistic Regression - Homework 2

The goal of this assignment is to create an artificial intelligence model that can correctly classify breast cancer as malignant or benign using logistic regression. We will take data from https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html and from this data, we will train a model using parts of the data, validate our results using a different part of the data, and then test our model on yet another part of the data.

I am expecting to see the model be able to correctly classify most of the tumors in the testing dataset after being trained and validated. We will be able to understand the effectiveness of our model using evaluation measures such as accuracy, precision, recall, and F1 score.

Derivations

Our Logistic Regression model will use the same matrix multiplication process as linear regression. The only difference is that we will put the result into a sigmoid function to have a percentage level of confidence instead of a continuous variable.

Linear Regression: $y = \theta^T x_b$

Logistic Regression: $y = \sigma(\theta^T x_b) \quad where \quad \sigma = \frac{1}{1+e^{-z}}$

Our Loss function is going to be a binary cross-entropy loss which is modeled by

$$L(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(h_i) + (1 - y_i)\log(1 - h_i)]$$

Our gradient derivation is

$$\frac{1}{n}X_b^T(h - y)$$

And our loss function that is using L2 Regularization is modeled by

$$L(\theta) + \frac{\lambda}{2n}\left|\left|\theta\right|\right|^2$$

Implementation

Like our Linear Regression model, many of the initial steps remain the same. We are using the same Python libraries as before, except this time, we also import some data from the dataset mentioned above. We then split the data into three categories with each category being a training, validation, and testing set respectively. Once we load and parse the data, we must standardize each category individually. We then set our hyperparameters, most of which are the same as in the first homework assignment, except for Lambda which we are using to regularize the data to prevent overfitting.

For the actual training, logistic regression uses the same methods as linear regression, but we place the results into a sigmoid function to make sure that each result is in the form of a percentage from [0-1]. We also add a regularization factor to the loss gradient. Then, we update our weights, keep track of the losses, and repeat this process for a large number of iterations. We can graph the loss gradient with losses we kept track of.

To evaluate the model, we use a set of evaluation variables. These will tell us how our model performed on the test data after being trained on the training data.

Evaluation & Results

Confusion matrix:

	Predicted Positive	Predicted Negative
Actual Positive	TP: 72	FN: 0
Actual Negative	FP: 2	TN: 41

Evaluation Metrics

Accuracy	0.9826086956521739
Precision	0.972972972972973
Recall	1.0
F1-Score	0.9863013698630138
TPR (Recall)	1.0
FPR	0.046511627906976744
ROC	≈ 0.98

The results of the model are outputted at the bottom of the Jupyter notebook file