2. Logistic Regression

The Link to the Notebook:

https://colab.research.google.com/drive/1Y-vnrZXdLTgSEptHxJV7tFpSXvPMn5Of?usp=sharing

The Link to the Repository:

https://github.com/yasinbastug/cmpe462_p1

This repo also contains the notebook(.ipynb) and the python(.py) file. You can also find the code for training and testing the logistic regression for the dataset in the 3th question.

You can run the python code with the following command:

```
python logreg.py
```

Getting Familiar with the Data

We fetch the data from ucimlrepo then load it into a pandas dataframe. We see that the data contains 8 columns and 3810 rows. The columns are as follows: Area, Perimeter, Major_Axis_Length, Minor_Axis_Length, Eccentricity, Convex_Area, Extent and Class

Data Preprocessing

We check for missing values and find none. We then check for the data types of the columns and find that all columns are of type float64 and int64 except for the Class column which is of type object. We then convert the Class column to 0 and 1 where 0 represents the 'Cammeo' class and 1 represents the 'Osmancik' class. We normalize the data using the MinMaxScaler. We then split the data into training and testing sets after shuffling by 4/1.

Methods for Logistic Regression

we have defined the following functions:

1. initialize_weights(dim): This function initializes the weights and bias to zero according to the dimension of the dataset.
2. sigmoid(z): This function returns the sigmoid of the input.
3. logloss(y_true, y_pred, epsilon=0): This function returns the log loss of the true and predicted values. Log loss is calculated as the negative of the mean of the sum of the true values multiplied by the log of the predicted values and the sum of the difference between 1 and the true values multiplied by the log of the difference between 1 and the predicted values. I added epsilon to avoid division by zero. We will use epsilon in the stochastic gradient descent function since y_pred can get so close to 0 or 1 that it can cause a division by zero.
4. gradient_dw(x, y, w, b, alpha, N): This function computes the negative of the gradient of the weights. The gradient of the weights is calculated as the product of the input and the difference between the

true value and the sigmoid of the dot product of the weights and the input. We also add the regularization term to the gradient of the weights. The regularization term is calculated as the product of the regularization parameter alpha, the weights squared, and the number of samples N. Since it is negative of the gradient, we subtract it from the gradient of the weights. We will be updating the weights by summing this gradient times learning rate in the gradient descent functions.
5. gradient_db(x, y, w, b): This function computes the negative of the gradient of the bias. The gradient of the bias is calculated as the difference between the true value and the sigmoid of the dot product of the weights and the input.

Training the Model

Training the model with batch gradient descent

For the logistic regression with batch gradient descent, we have defined the train_batch_gd function. This function takes the input features X, the true values y, the weights w, the bias b, the regularization parameter alpha, the learning rate lr, and the number of epochs. The function initializes the previous loss to infinity and the start time to the current time. It then initializes an empty list to store the losses for each epoch. The function then iterates over the number of epochs. In each epoch, the function calculates the gradient of the weights and the bias for the whole dataset and takes the mean of the gradient set. The function then updates the weights and the bias for the whole dataset. The function then calculates the predictions and the loss for the entire dataset. The function then appends the loss to the losses list. The function then prints the epoch and the loss. There is a stopping condition that finishes the algorithm if the loss is not decreasing, the function stops training and prints a message. The function then calculates the end time and the training time. The function then returns the weights, the bias, the losses, and the training time. Training time and the loss list are for plotting.

Training the model with stochastic gradient descent

For the logistic regression with stochastic gradient descent, we have defined the train_sgd function. This function's difference from the batch one is that we update the weight and bias for each instance in the dataset. The function shuffles the dataset and iterates over each instance in the dataset. The function calculates the gradient of the weights and the bias for each instance and updates the weights and the bias. The function then calculates the predictions and the loss for the shuffled instance. The function then appends the loss to the epoch_losses list. The function then calculates the average loss for the epoch and appends it to the losses list. The function then prints the epoch and the average loss. There is a stopping condition that finishes the algorithm if the average loss is not decreasing, the function stops training and prints a message. The function then calculates the end time and the training time. The function then returns the weights, the bias, the losses, and the training time. Training time and the loss list are for plotting.

Testing the Model

Loss Calculation

The test function takes the input features X, the true values y, the weights w, and the bias b. The function initializes an empty list to store the predictions. The function then iterates over the input features and calculates the sigmoid of the dot product of the weights and the input features and the bias. The function then appends the sigmoid value to the predictions list. The function then calculates the log loss of the true values and the predictions. The function then returns the loss.

Prediction for Accuracy Calculation

The predict function takes the input features X, the weights w, and the bias b. The function initializes an empty list to store the predictions. The function then iterates over the input features and calculates the sigmoid of the dot product of the weights and the input features and the bias. If the sigmoid value is greater than 0.5, the function appends 1 to the predictions list; otherwise, it appends 0. The function then returns the predictions.

Analysis

2. 5-fold cross-validation

We shuffled the data and split it into 5 folds. First for the batch gradient descent, tested 5 different lambda values for the regularization parameter: 0.01, 0.1, 1, 10, 100. We trained and tested for all the folds and the final losses and accuracies turned out to be as follows:

Lambda	Loss	Accuracy
0.01	0.407	0.902
0.1	0.391	0.922
1	0.383	0.941
10	0.330	0.928
100	0.546	0.654

as we can see the lowest loss is for lambda = 10 and the highest accuracy is for lambda = 1. So we tested for both values with 400 epochs, lambda = 1 seemed to be performing better with a lower loss and higher accuracy. So we chose lambda = 1 for the final batch logistic regression model.

For the stochastic gradient descent, we again tested 5 different lambda values for the regularization parameter: 0.01, 0.1, 1, 10, 100. And 0.1 was the best performer.

Lambda	Loss	Accuracy
0.01	0.191	0.895
0.1	0.185	0.928
1	nan	0.458
10	nan	0.451
100	nan	0.425

As you can see the loss diverges for lambda = 1, 10, 100. So we chose lambda = 0.1 for the final stochastic logistic regression model.

3. Performances of Stochastic and Batch Logistic Regression with and without Regularization

The results for the final model are as follows:

Method	LR	Epoch	Training Accuracy	Test Accuracy
GD	1	400	0.921	0.912
L2-GD	1	400	0.922	0.912
SGD	0.1	<10	0.932	0.917
L2-SGD	0.1	<10	0.930	0.909

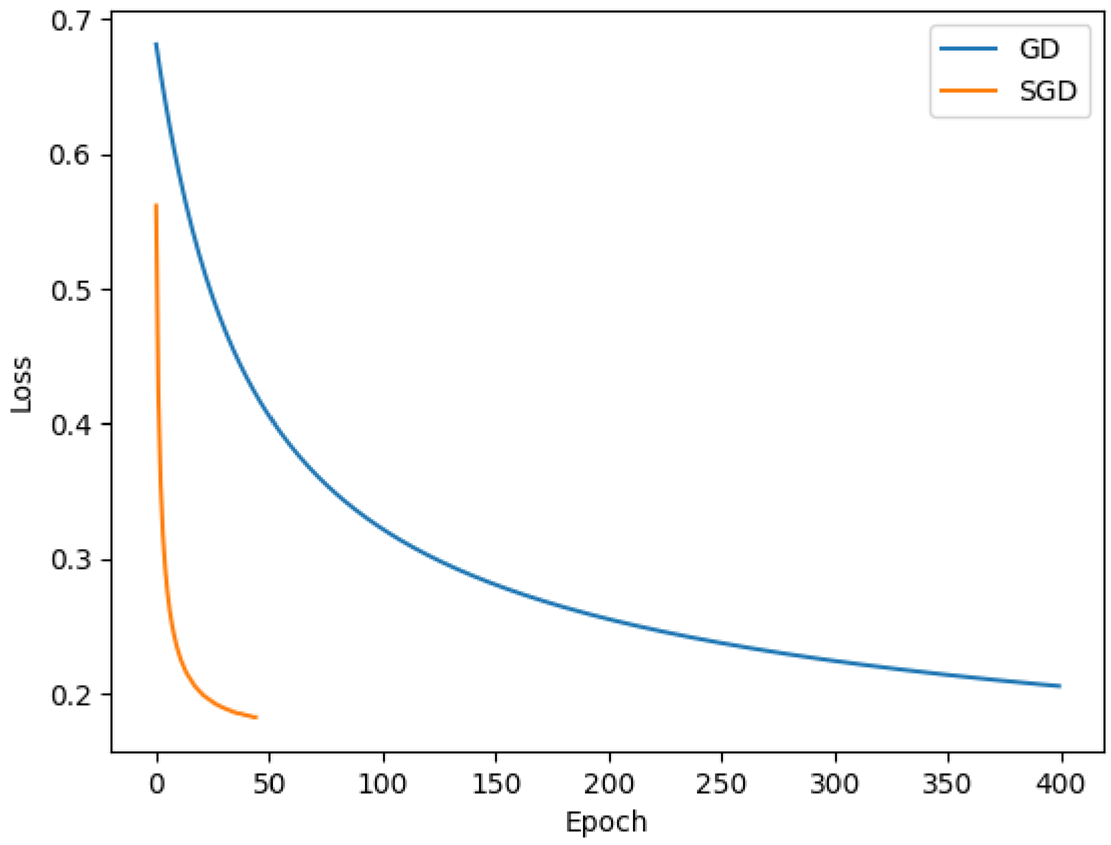
Comments on the Results

- For the batch gradient descent, the model with regularization performed nearly same with the model without regularization. I believe that we cannot observe the effect of the regularization term on the test dataset since it is not very large.
- For the stochastic gradient descent, the model with regularization performed worse than the model without regularization. We can observe the effect here since the regularization is effects on every instance iteration/weight update instead of every epoch in batch one. At the end we want this model to be able to predict well for unseen data. The regularization term prevents the model to overfit the data.

4. Comparison of the Training Times

Method	Training Time	Accuracy
GD	25.075	0.916
SGD	12.147	0.916

The plot for the losses by epoch for both models is as follows:



I chose the learning rate of 0.01 for the stochastic model to achieve similar accuracies with the batch one. We can see that the stochastic gradient descent converges faster than the batch gradient descent. This is expected since the batch gradient descent calculates the gradient for the whole dataset and updates the weights and the bias for the whole dataset. The stochastic gradient descent calculates the gradient for each instance and updates the weights and the bias for each instance. This makes the stochastic gradient descent faster than the batch gradient descent.

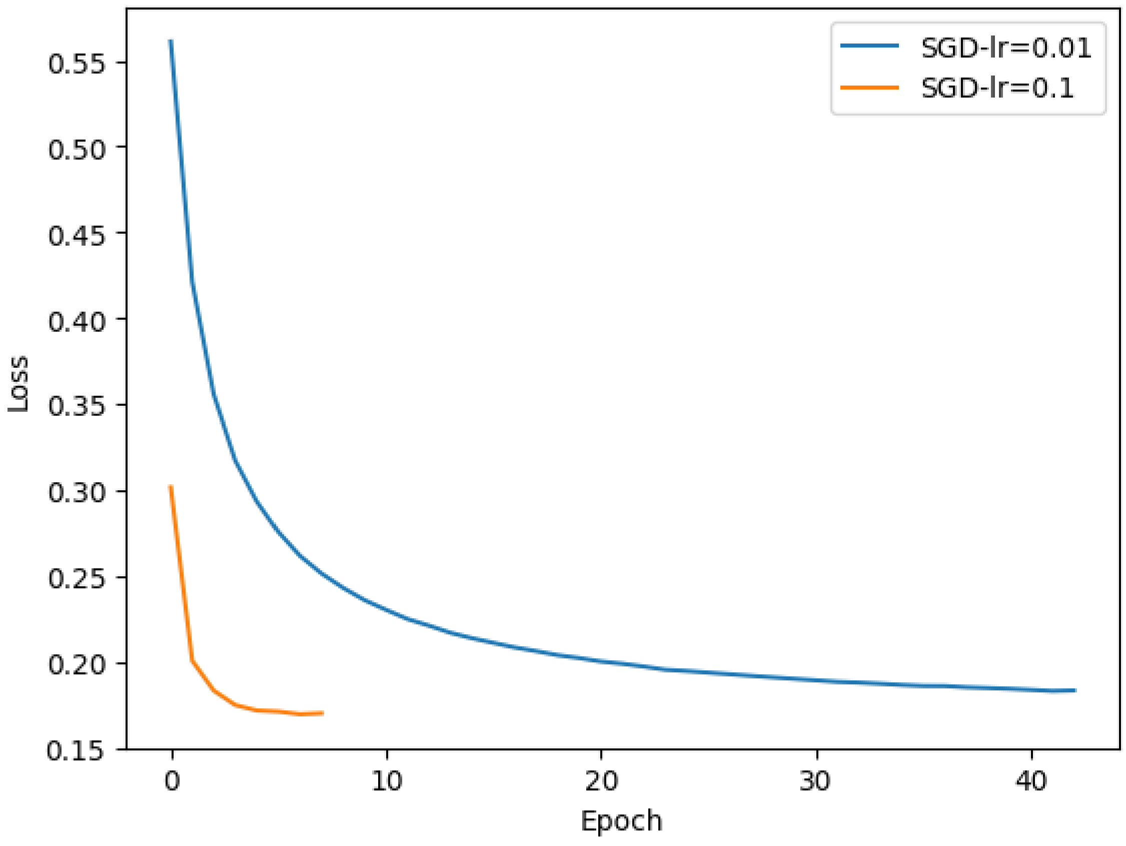
Investigation on Initial Step Sizes in SGD

For learning rates >0.1 the loss diverged and for learning rates <=0.1 the loss converged. We then tested 0.1 and 0.01 specifically.

The results for the final model are as follows:

Learning Rate	Training Time	Accuracy
0.01	8.328	0.917
0.1	1.922	0.916

The plot for the losses by epoch for both models is as follows:



We can see that the learning rate of 0.1 converges faster than the learning rate of 0.01. The accuracy of 0.01 is slightly better than the 0.1, this is because the learning rate of 0.1 is higher and the model overshoots the minimum. The learning rate of 0.01 is more stable and converges to the minimum. But the difference in accuracy is not significant since the time difference is very high. So we can choose the learning rate of 0.1 for the final model.

Resources

https://heena-sharma.medium.com/logistic-regression-python-implementation-from-scratch-without-using-sklearn-d3fca7d3dae7#do7d

https://machinelearningmastery.com/scale-machine-learning-data-scratch-python/

https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/

https://www.kaggle.com/code/sagira/logistic-regression-math-behind-without-sklearn

https://realpython.com/gradient-descent-algorithm-python/#:~:text=Stochastic%20gradient%20descent%20is%20an,used%20in%20machine%20learning%20applications.

https://machinelearningmastery.com/k-fold-cross-validation/

Copilot, ChatGPT